UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

MÁSTER UNIVERSITARIO
DE INGENIERÍA DE ORGANIZACIÓN

ESTUDIO PARA EL SECUENCIAMIENTO DE TAREAS INDEPENDIENTES EN ENTORNOS HETEROGÉNEOS

Departamento de Sistemas, Automática
e Informática Industrial

Autor: Ángel Santos Guerrero
Director: Antoni Guasch Petit

Junio 2016

TRABAJO FINAL DE MÁSTER

Agradecimientos

A Isabel por su apoyo incondicional
durante largas horas de dedicación a este proyecto.

A Daniel por sus ya 13 meses de incontables sonrisas

A papá y mamá, por motivos obvios

A Toni Guasch por su seguimiento
y soporte durante este TFM

RESUMEN

El presente trabajo de final de máster tiene por objetivo el estudio de la aplicación de distintas técnicas de planificación sobre un modelo de problema especificado por el departamento de Sistemas, Automática e Informática Industrial (ESAII).

El problema es de secuenciación de tareas en entornos heterogéneos, con la especificidad de ser trabajos con llegadas dinámicas al sistema. Existe mucho conocimiento sobre metodologías de planificación de tareas, cuando la planificación es estática (se dispone de toda la información antes de iniciar el proceso de planificación) y también cuando es dinámica y se dispone igualmente de la información necesaria para poder ejecutar las planificaciones.

En este caso de estudio, las llegadas de las solicitudes de trabajo a planificar no son conocidas ni previsibles, así como tampoco son conocidos los tiempos de proceso de cada trabajo que se solicite al sistema. Además el sistema es un sistema Soft Real-Time, por lo que es necesario tener en cuenta que debemos planificar con el objetivo de respetar los plazos máximos de entrega que se nos especifica.

Dentro de este marco de trabajo, se evaluarán los resultados de aplicar 3 diferentes técnicas de planificación existentes en la literatura y más reconocidas, para extraer conclusiones sobre cuál de ellas puede ser más recomendable: Earliest Deadlines First, Least Laxity First y Myopic. Estas tres técnicas se tendrán que complementar con la gestión de estimaciones de tiempos de proceso como explicaremos en esta memoria.

De estas 3 posibilidades, llegaremos a la conclusión que, teniendo en cuenta la gran influencia de no disponer de tiempos de proceso conocidos, parece más efectiva la metodología de Earliest Deadline First, por ser la más sencilla y por no gestionar las prioridades en base a tiempos de proceso no conocidos (o tiempos de proceso estimados).

Finalmente, de los resultados de este estudio, se podrán extraer tres posibles caminos por los que se podría ampliar esta investigación para poder mejorar los resultados obtenidos con estas técnicas clásicas.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE FIGURAS

1. INTRODUCCIÓN

La teoría de la planificación de tareas es actualmente de aplicación generalizada en la industria. Es un campo en que las investigaciones empezaron hace más de 40 años y que está en continua evolución. Las numerosas líneas de investigación que existen son debidas a los cambios constantes en la estructuración y necesidades de la industria, y también en gran medida, a que las herramientas de cálculo van incrementando su capacidad técnica y permiten abordar soluciones que anteriormente no eran viables por las limitaciones del *hardware*.

La importancia de la planificación ha ido creciendo conforme lo ha hecho la presión de costes en la industria, proporcional al incremento de la competitividad de los mercados. Todo esto hace necesario el apoyo de una buena planificación de la producción que permita optimizar los costes (o minimizar los sobrecostes consecuencia de la producción). Entre los costes más estándares están tanto los costes de almacenaje, como consecuencia de fabricar demasiado pronto, como los costes por entrega tardía, como consecuencia de haber fabricado demasiado tarde. Otros costes, no menos importantes, son el de la maquinaria, así como otros más difícilmente cuantificables, como son la insatisfacción de los clientes o la pérdida de reputación hacia ellos, ambos fruto posiblemente tanto de entregas tardías como de costes demasiado elevados. Por ello, en los inicios de la teoría de la planificación, el problema más común que se quería resolver era: "Cómo planificar las tareas de forma a que con un nº de máquinas definido (recurso limitado), podamos dar respuesta a la producción dentro de las fechas solicitadas, minimizando los costes, tanto de maquinaria, almacenaje y penalizaciones por entrega tardía".

La teoría de la planificación de tareas se descompone en **planificación estática** y en **planificación dinámica**. La aplicación de un tipo o de otro depende del tipo de sector industrial que se esté considerando. Así, una planificación estática es adecuada cuando se conocen todos los datos de las tareas a planificar en el momento inicial antes de comenzar los procesos, y una planificación dinámica es la que debe aplicarse cuando la llegada de las tareas es dinámica o simplemente, no es posible conocer de antemano, la duración de las tareas en proceso. En el primer caso puede asociarse fácilmente con la filosofía "Just In Time" de la industria de fabricación, y en el segundo caso, puede aplicarse tanto a tareas realizadas por procesadores de ordenador, como a centralitas de llamadas, siendo las llamadas como tareas, y los operadores como máquinas o procesadores, dado que en ambos casos, son llegadas

de tareas en instantes no definidos o conocidos, así como que tampoco se conoce la duración de las mismas.

El presente trabajo de final de máster en Ingeniería de Organización de la Universidad Politécnica de Catalunya, queda enmarcado dentro de la *planificación dinámica*, para la *secuenciación de tareas independientes en entornos heterogéneos*. Pretende llegar a unos resultados que permitan establecer una comparativa de rendimiento de distintos métodos dinámicos ***EDF*** *(Earliest Deadline First),* ***LLF*** *(Least Laxity Time) y* ***Myopic***, aplicados sobre un caso que contempla un dimensionamiento del número de tareas a planificar, así como en número de máquinas (o servidores), mucho mayor del que se viene estudiando en la literatura disponible. Estos métodos se describen con más detalle en el apartado 4 y 7 de la presente memoria.

El entorno en el que se presenta este proyecto, por el número de tareas a planificar así como su duración corresponde a *sistemas de tiempo real multiprocesador*. En nuestro caso estaremos considerando servidores multiprocesador. En el apartado 5 de esta memoria se describe este sistema con más detalle.

2. MOTIVACIÓN

(Justificación de la utilidad y necesidad del TFM)

Este proyecto viene motivado fundamentalmente por una necesidad del departamento. Quedará enmarcado dentro de un proyecto de mucha mayor magnitud, que ha sido aprobado recientemente, de forma a poder dar un soporte didáctico y académico adicional con nuestros resultados. Este proyecto pretende generar información o dar visibilidad sobre los resultados de una *planificación dinámica* cuando disponemos de *n tareas* a distribuir entre *m servidores* con *k procesadores* cada uno, con llegadas estocásticas y tiempos de proceso aleatorios (con *n* muy grande).

Tras las simulaciones, que se plantearán más adelante, dispondremos de información suficiente para poder comparar el comportamiento de los 3 algoritmos de aplicación en entornos de sistemas en tiempo real multiprocesador como el que se nos presenta. La comparativa será realizada sobre los algoritmos dinámicos del tipo ***EDF*** *(Earliest Deadline First),* ***LLF*** *(Least Laxity Time) y* ***Myopic***, en base a los indicadores seleccionados que describiremos en el apartado 7 de esta memoria.

La herramienta que utilizaremos, ***ARENA*** de **Rockwell Automation**, nos facilitará la tarea para poder implementar el código de los algoritmos seleccionados y poder recoger los datos para ser estudiados posteriormente. Uno de los puntos fuertes es que ***ARENA*** nos permitirá utilizar entradas de datos en base a distribuciones de muestras reales, además de contar con la simulación y ejecución de algoritmos en tiempo acelerado. De esta forma, los resultados extraídos de las simulaciones corresponderán a resultados en base a datos de entrada reales, y por lo tanto, podrán ser tomados como previsiones reales. Por el contrario, el uso de ***ARENA*** bajo *licencia académica* nos obligará a utilizar un modelo escalado para poder ser utilizado bajo dicha licencia. Este aspecto deberá ser tenido en cuenta en cuanto a la aplicación de los resultados finales sobre el modelo real, y deberá valorarse en tal caso, la necesidad de uso de *licencia profesional* de dicha aplicación.

3. OBJETIVOS
(Objetivo del Trabajo de Final de Máster)

Los objetivos de este trabajo son estudiar, evaluar y analizar los algoritmos de secuenciación o planificación de tareas independientes en entornos heterogéneos. Como ya hemos comentado anteriormente, el problema aparece en procesos productivos, en procesos de servicios como por ejemplo los "call centers", o en procesos de origen logístico, como puede ser la gestión de empresas de venta de suministros online.

En una fase inicial, se realizará una valoración de la literatura existente sobre la *planificación dinámica* y algoritmos de secuenciación existentes. En este apartado, también se dedicará un espacio para describir la *planificación estática*, dado que en caso de secuenciación por lotes, cabe la posibilidad de aplicar algoritmos que ya están consolidados dentro de la planificación estática.

Tras esta valoración inicial general, se seleccionarán los algoritmos candidatos que se ajusten más a las necesidades de nuestro planteamiento. Recordemos que se trata de un número muy grande de tareas para un número también grande de servidores y procesadores. Teniendo en cuenta que también se especifica que la tasa de llegadas será alta, deberemos seleccionar algoritmos que nos permitan soluciones rápidas para evitar bloqueos de sistema por motivos de carga de cálculo. Aparte de este requisito de velocidad de cálculo, deberemos focalizar igualmente, hacia algoritmos con prestaciones adecuadas y minimizar el número de tareas que queden fuera de plazo de servicio.

Una vez tengamos estudiados, evaluados, seleccionados y programados los algoritmos, recogeremos los datos para realizar los análisis comparativos entre los diversos algoritmos, y presentar las conclusiones que se puedan derivar de estos resultados.

4. ANÁLISIS BIBLIOGRÁFICO
(Antecedentes previos y Estado del arte)

En el ámbito de planificación de trabajos o tareas para servidores, se pueden encontrar dos textos de referencia que nos describen a la perfección el estado del arte actual, a partir de los cuáles podemos decidir profundizar en una dirección u en otra en función de las inquietudes a las que queremos atender en este Trabajo Final de Máster(TFM).

En el caso de Ruby Anenette y Aisha Banu[2] se nos presenta una revisión de la planificación de trabajos aplicada al '*cloud computing'* o trabajo en la nube. Esta vía nos permite escalabilidad en la disponibilidad de recursos para ejecutar los trabajos necesarios, en función de la demanda de los usuarios. La gestión de la planificación de tareas nos facilita controlar el coste y operar de una forma optimizada.

El uso actual que hacemos de las tecnologías, junto con la evolución de las conexiones de internet y aumento de los anchos de banda, hace que se incremente el uso de este '*cloud computing'*.

Por otro lado Lindh y Otnes[1] presentan un repaso sobre los algoritmos de planificación más representativos utilizados en el ámbito de los sistemas en tiempo real, tanto para los casos de monoprocesador como multiprocesador. Lindh y Otnes[1] definen '*algoritmo de planificación*' como un conjunto de reglas o normas que el sistema de planificación debe seguir para gestionar el sistema de tiempo real. Es decir, se fija la forma de gestionar la cola de tareas y autoriza el uso de tiempo de procesador para dar servicio a las tareas.

En el marco de nuestro TFM, es decir, y considerando que las tareas son independientes entre sí, los algoritmos de planificación se pueden clasificar en '**estáticos**' y en '**dinámicos**', en función del momento en el que se realiza la planificación. Para una **planificación estática**, se entiende que todas las tareas que deben ser planificadas, así como toda la información que le afecta, y la disponibilidad de los recursos, están disponibles en el momento en el que se realiza la planificación. Mientras, en una **planificación dinámica**, no se conocen de antemano los recursos necesitados las tareas, dado que las tareas llegan y la planificación se realiza en tiempo real.

Otras clasificaciones de los sistemas de planificación son las siguientes:

1. **Hard Real-Time / Soft Real-Time**: Se define en función de la rigidez de los plazos de entrega solicitados a cada tarea y se entiende como sistema **Hard Real-Time** aquel en que el cumplimiento de los plazos de entrega es crítico. En caso contrario, se producirá un fallo en el sistema irrecuperable, impidiendo al sistema seguir operando. Por el contrario, en el caso de trabajar con un sistema **Soft Real-Time**, aunque también se basa en el cumplimiento de los plazos de entrega de las tareas, no se produce fallo de sistema en los casos en que la entrega de las tareas se realice fuera de plazo. En esos casos, el sistema tendrá menores prestaciones, pero no dejará de operar.
2. **Expulsor / No Expulsor**: Se define un sistema como **Expulsor** cuando permite que una tarea con entrada en el sistema más reciente y más prioritaria que la que se esté atendiendo pueda disponer del recurso de forma inmediata para ser atendida. La tarea que estaba en curso quedará interrumpida y a la espera de disponer de nuevo del recurso para poder proseguir con su ejecución. El sistema se considerará como **No Expulsor** cuando se determina que una tarea cuya ejecución ha comenzado, no libera el recurso hasta que no es finalizada. Este caso, no se permite entonces la interrupción de la ejecución de tareas, una vez iniciadas.
3. **Periódico / Aperiódico**: Un sistema **periódico** ejecuta tareas periódicas, dónde el periodo de cada tarea es factor determinante a respetar. En el caso de que las tareas que se ejecuten no sean periódicas, se tratará de un sistema **aperiódico**.
4. **Monoprocesador / Multiprocesador**: Otra posible clasificación es por la tipología de procesador. Un sistema se dice **Monoprocesador** cuando utiliza un único recurso o procesador para la ejecución de las tareas. En el caso de que el sistema disponga de varios procesadores, éste se dirá **Multiprocesador**, permitiendo el uso de los recursos en paralelo, y por lo tanto una mayor capacidad de cálculo disponible.

Es necesario notar, de forma general, que los métodos exactos no son contemplados en este marco de planificación de tareas en servidores, debido a que el elevado coste computacional hace que su uso sea inviable. En el caso de planificación dinámica, no es posible garantizar siquiera que la planificación resultante llegue a tiempo de ser implementada.

Dentro de la planificación estática, los algoritmos son basados en *Heurísticas* o bien *Guided Random Search*. En ambos tipos, la solución que se propone no es óptima, pero si es suficientemente buena como para resultar en un coste muy reducido. Los métodos Guided Random Search brindan mejores soluciones que las Heurísticas simples, pero, en contrapartida, su coste computacional es mayor.

Por otro lado, los algoritmos de planificación dinámica se dividen a su vez en *planificación por lotes* (batch mode) y en *planificación on-line*. En el caso de *planificación on-line*, una tarea es planificada en un servidor a su llegada al sistema. Se entiende, en general, que una vez se ha planificado, ya no se podrá modificar su planificación. Es una buena estrategia cuando la tasa de llegadas es baja. Por contra, para una *planificación por lotes*, se planifican conjuntos de tareas, con la característica de que en cada proceso de planificación por lotes, se pueden utilizar todas las tareas (aunque hayan sido previamente planificadas) que no hayan empezado su ejecución en procesador.

Bien sea para el caso de planificación estática o bien para planificación dinámica, los algoritmos pueden pertenecer al grupo de los que optimizan el uso de recursos para dar respuesta a todas las tareas o bien al grupo de los que realizan la planificación para dar respuesta a todas las tareas dentro del plazo máximo que se le asigna a cada una inicialmente.

En el cuadro (figura 4.1) que se presenta a continuación pueden verse distribuidos los métodos estáticos y dinámicos más representativos en función de los parámetros que se optimizan con su uso. Concretamente existen dos grandes grupos; uno de ellos potencia la optimización de los recursos (intenta maximizar su utilización), a la vez que optimizar el coste ocasionado. El segundo grupo planifica para maximizar el número de tareas entregadas dentro del plazo solicitado al inicio.

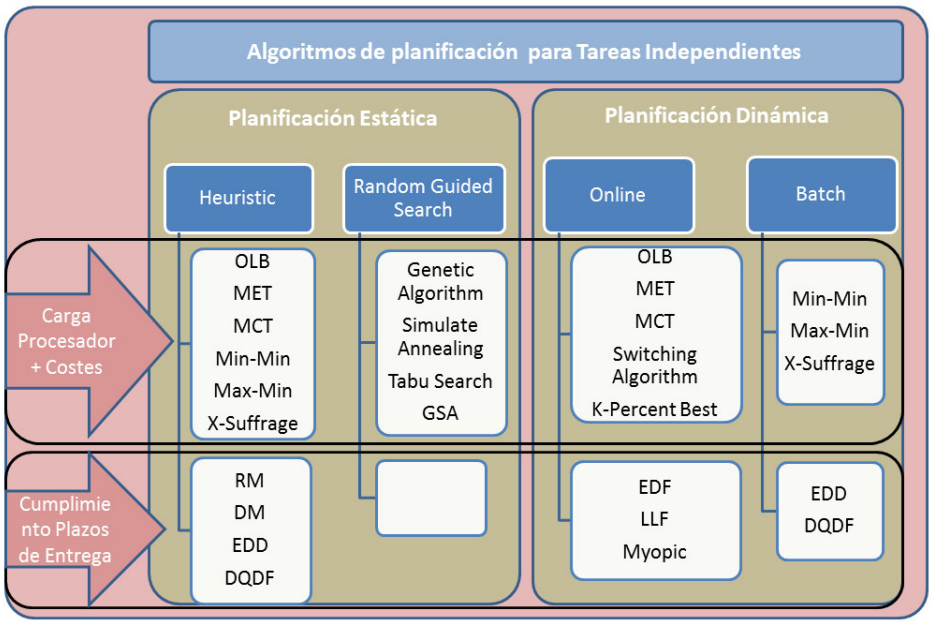


Figura 4.1

4.1 Algoritmos para planificación estática

Describiremos brevemente los algoritmos heurísticos de **planificación estática**, para tareas independientes:

- *Oportunistic Load Balancing (OLB)* [7]

  Su objetivo es mantener los recursos lo más ocupados posible, por lo que se trata de una asignación rápida a recursos disponibles siguiendo un orden cualquiera, sin tener en cuenta ninguna duración de tareas en curso. Es un método muy sencillo, dado que no requiere de cálculo, pero nada optimizado en cuanto a tiempo de ocupación de recurso.

- *Minimum Execution Time (MET)*

  En este caso, se pretende asignar cada tarea en el servidor que menor tiempo de proceso necesite para ejecutarla, sin tener en cuenta la disponibilidad del recurso [7]. Ignorar la disponibilidad de los servidores durante la asignación de recursos, puede producir descompensación en el reparto de carga entre los procesadores o recursos.

- *Minimum Completion Time (MCT)*

  Este método intenta utilizar los aspectos positivos de los dos anteriores. Asignará la tarea al servidor que antes pueda acabarla [7]. Es importante destacar que se van asignando las tareas por orden arbitrario. El problema de poder generar una descompensación de carga entre servidores sigue presente, puesto que cuando se calcula el *tiempo mínimo de entrega*, no se tienen en cuenta las tareas que puedan haber a la espera del recurso.

- *Heurística min-min*

  Este método calcula el tiempo mínimo de entrega de cada tarea y asigna la tarea con el menor de ellos al servidor que le corresponda. Una vez hecho, se retira la tarea del lote a planificar y el servidor al que ha sido asignado, y se repite el procedimiento con las tareas restantes hasta que todas son asignadas [8]

  La ventaja de este método es que asigna carga a los recursos en la menor cuantía posible a medida que va asignando las tareas. Esto permite tener mayor margen o capacidad disponible para asignar más tareas.

- *Heurística max-min*

  Las tareas con mayor tiempo de ejecución se asignan primero al mejor servidor para ellas. Este método ayuda a minimizar las penalizaciones en las que se incurren cuando asignamos las tareas más largas en último lugar. En esos casos, es probable que lleguen fuera de plazo.

- *Xsuffrage*

  La tarea que se vea más perjudicada por una no asignación es la que debe ser asignada prioritariamente; ésta es la base del método Suffrage [9]. El perjuicio se evalúa en función del valor MCT de cada tarea. Xsufferage[10] es una mejora respecto de Suffrage que tiene en cuenta el caso de agrupación de recursos.

En cuanto a los algoritmos de búsqueda aleatoria, estos son más complejos que los precedentes, y tenemos entre los más representativos:

- *Genetic Algorithm (GA)*

  Este algoritmo consta de 4 etapas (Seleccionar, Cruzar, Mutar y Reproducir) hasta llegar a un conjunto de soluciones para la planificación. Su ventaja es la obtención de unas soluciones posibles de buena calidad. No obstante, el tiempo de cálculo necesario es muy superior a los métodos más sencillos expuestos más arriba.

- *Simulate Annealing (SA)*

  Este método [11] intenta encontrar la solución (o distribución de tareas entre los recursos disponibles) para conseguir una distribución óptima de carga entre los recursos, con un *tiempo de entrega* (o finalización) mínimo.

- *Genetic Simulated Annealing (GSA)*

  Esta heurística es una combinación de GA y SA. Se utiliza el 'motor' del algoritmo genético, pero para la etapa de selección, se utiliza el algoritmo Simulate Annealing

4.2 Algoritmos para planificación dinámica

Respecto de los métodos de **planificación dinámica**, las posibilidades para planificación estática con OLB, MET o MCT pueden ser aplicadas también en una planificación dinámica *on-line*.

Existen también otros algoritmos reconocidos como pueden ser Switching Algorithm (SA) o k-percent best (KPB) que se expone a continuación de forma breve.

- *Switching Algorithm (SA)*

Este algoritmo hace un uso combinado de los métodos MET y MCT. MET es usado para utilizar el servidor más rápido, hasta llegar a un umbral (fijado por sistema o por el administrador del mismo) de ocupación de servidores, a partir del cual, pasaría a utilizarse MCT como técnica de asignación de tarea a un servidor. Este último paso es el que permitiría compensar la ocupación entre los distintos servidores, que puede generar el uso del algoritmo MET.

- *K-Percent Best (KPB)*

Este algoritmo consiste en definir conjuntos de k-servidores asignables a tipos de tareas. De esta forma, se pueden reservar una serie de servidores para tareas que se estiman más críticas y que deban ser atendidas nada más llegar al sistema. Como criterio general, los k-servidores seleccionados son los k mejores entre los servidores disponibles para el tipo de tarea al que se preasignan.

Los métodos presentados hasta este momento son de difícil aplicación a nuestro caso, puesto que:

- Son esencialmente utilizados para planificación estática, aunque en algunos casos se podrían utilizar en caso de planificación dinámica por lotes ('*batch mode*')
- Ninguno de ellos tiene en cuenta la solicitud de plazo máximo de entrega que debemos cumplir, ya que están fundamentalmente basados en optimizar la carga de trabajo de los servidores, para minimizar el intervalo de tiempo necesario para finalizar las tareas ('*makespan*'), y el coste por la planificación en sí.
- Todos ellos necesitan conocer los tiempos de proceso exactos para poder ser utilizados. En nuestro caso de estudio, no disponemos de este dato, y

tendremos que utilizar, en caso necesario, un histórico de las medias de tiempos de proceso que tengamos a medida que transcurra la simulación.

- En el caso de búsqueda aleatoria, en nuestro caso de estudio, su utilidad es dudosa, puesto que además de partir de una solución propuesta aproximada, los cálculos se realizarían con datos de tiempos de proceso que no conocemos. El resultado podría estar muy lejos de la realidad. No se recomienda su uso por lo tanto para nuestro TFM.

Por estos motivos anteriores, es más adecuado utilizar métodos de planificación dinámica online para el caso de estudio que estamos considerando.

4.3 Planificación para sistemas de tiempo real

En los sistemas de tiempo real, ámbito en el que se enmarca el presente TFM, las tareas tienen un **plazo de entrega**, un **tiempo de ejecución** y un **instante de tiempo de llegada al sistema**. El plazo de entrega, hasta este momento, no ha sido considerado como restricción en ninguno de los métodos descritos, tanto en el apartado 4.1 como en el apartado 4.2.

Sobre las tareas, tal y como son caracterizadas en un sistema de tiempo real, debemos presentar las siguientes definiciones que serán utilizadas a lo largo de la presente memoria:

Tiempo de llegada al sistema (*Release time*): Instante de tiempo en el que la tarea está disponible para iniciar su ejecución. En nuestro caso de estudio, este instante corresponde con el instante de tiempo en que llega al sistema.

Plazo de entrega máximo: Es el instante de tiempo máximo, antes del cual la tarea debe estar finalizada.

Tiempo de ejecución: Es el tiempo de procesador necesario para ejecutar la tarea.

Para un **sistema monoprocesador**, los algoritmos más representativos para planificación dinámica, aplicados on-line, y considerando los plazos de entrega como restricción, son:

- *Earliest Deadline First (EDF)* [12][13]

Es un algoritmo que fija la prioridad de asignación en función del plazo de entrega de cada tarea. La tarea más prioritaria es aquella que tiene un plazo de entrega

más urgente. Este tipo de prioridad es una prioridad dinámica, dado que se asigna a la llegada de cada tarea al sistema.

- *Least Laxity First (LLF)* [12][13]

Es un algoritmo que fija la prioridad de asignación en función de la holgura de cada tarea. Por *holgura* entenderemos

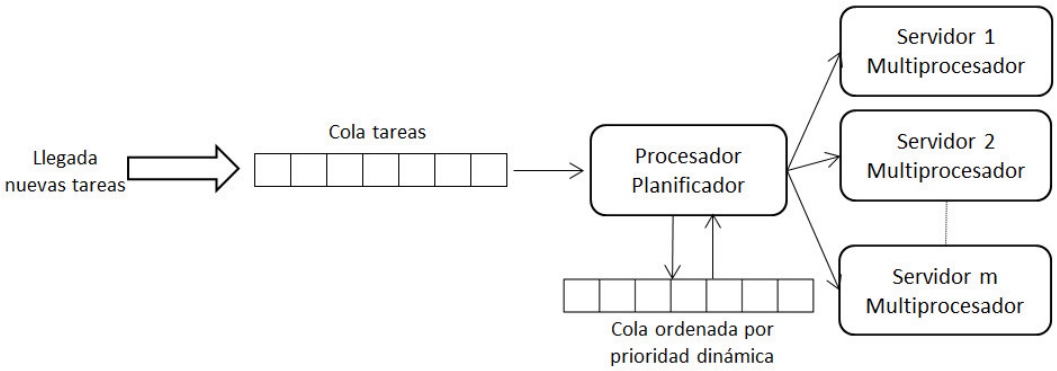
$$Holgura = Plazo\ Entrega\ Máximo - Tiempo\ de\ Ejecución$$

Aquella tarea que tenga la menor holgura respecto de las demás, será la más prioritaria en el momento de asignación. Este tipo de prioridad también es dinámica, dado que cambia en función de las llegadas de tareas al sistema.

Para los **entornos multiprocesador** [1], cualquier algoritmo desarrollado para monoprocesador puede ser aplicado, como EDF o LLF, gestionando cada procesador como si se tratase de un sistema monoprocesador. No obstante, la gestión y seguimiento de la ejecución de la planificación en estos casos se vuelve muy compleja, por lo se disponen habitualmente de heurísticas adicionales que permiten simplificar la tarea de encontrar una planificación factible. Los dos enfoques principales para aplicar heurística son **Planificación Global** y **Planificación Distribuida**.

- *Algoritmo de Planificación Global*

El algoritmo dispone de una cola con las tareas que están disponibles o preparadas para ser ejecutadas (ver figura 4.2), correspondiendo la posición en cola con su prioridad. El algoritmo planificador asignará la tarea en primera posición de la cola (será la más prioritaria) al procesador que primero queda disponible.



- *Algoritmo de Planificación Distribuida*

Este algoritmo facilita mucho la gestión inicial, puesto que cada tarea se asigna a un procesador en exclusiva y sólo será ejecutada en el procesador asignado (ver figura 4.3). Esta distribución hace resultar la planificación de un multiprocesador en un conjunto de planificaciones de monoprocesador. En estos casos, el reparto de las tareas condiciona la factibilidad de encontrar una planificación viable, puesto que este reparto tarea vs procesador [illegible] difícil de resolver. Habitualmente se solventa con técnicas heurísticas, que, aunque nos facilitan soluciones buenas, sabemos que no son las óptimas.

Figura 4.2

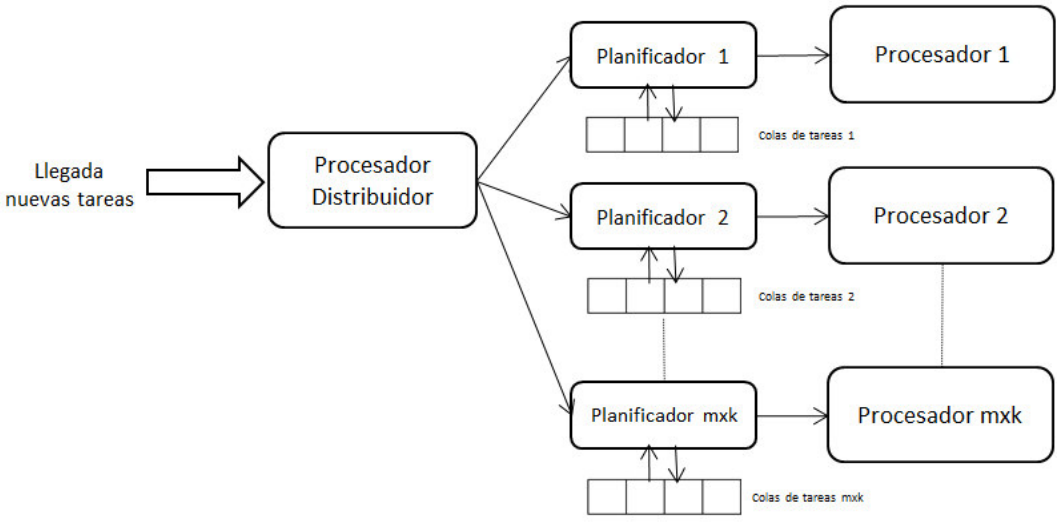


Figura 4.3

Aparte del uso de algoritmos inicialmente desarrollados para sistemas monoprocesador, se desarrolló en 1989 Ramamritham, Stankovic[4], un algoritmo específico para entorno multiprocesador:

- *Myopic Algorithm* [4]

Es un algoritmo de planificación específico para sistemas en tiempo real multiprocesador y puede, además de planificar tareas en función de la disponibilidad de tiempo, tener en consideración la necesidad de recursos específicos para la tarea. Utiliza métodos heurísticos para obtener una planificación viable: Cuando llega una tarea al sistema, se identifica cuál es la tarea más prioritaria de todas las disponibles

pendiente de planificar y se entrega a un procesador o servidor, que comprobará si la puede ubicar en su carga de trabajo de forma que la planificación que posee continúe siendo viable, incluyendo esta nueva tarea. En caso positivo, el método exige verificar que es '*Strongly-feasible'*. Para que una planificación sea '*Strongly-feasible*' debe cumplirse que, ampliando la planificación con las tareas que hubieran pendientes de planificar, la planificación siga siendo viable. Si la planificación es '*Strongly-feasible*', se siguen añadiendo tareas hasta completar la planificación. Por el contrario, en el caso de que no sea '*Strongly-feasible'*, deberá desestimarse la planificación encontrada y continuar buscando en el servidor siguiente, con la misma metodología.

Cuando una tarea llega al sistema y queda a la espera de ser ejecutada, el algoritmo utilizará las variables siguientes:

- T_A : Instante de llegada al sistema
- T_D : Plazo máximo de entrega o de finalización de ejecución
- T_C : Tiempo de ejecución máximo (peor caso) … *Worst Case Execution Time*
- T_R : Necesidad de recursos específicos

Las tareas, para este método, se entienden como **independientes**, **aperiódicas**, **no expulsoras** y usuarias de recursos de forma compartida o exclusiva. El conjunto de tareas pendientes de planificar estarán priorizadas en función de su T_D siendo la tarea que tenga menor T_D la más prioritaria. Cuando se planifica una tarea, el algoritmo calcula el instante de tiempo más temprano en que podrá ejecutar la tarea (T_{EST}), basándose en la disponibilidad de los recursos que le serán necesarios para su ejecución. Utilizando esta última definición, la siguiente condición tendrá que ser cierta para cada tarea cuando se implemente una nueva planificación:

$$0< T_A < T_{EST} < (T_D-T_C)$$

El cálculo, para este método, de T_{EST}, se realiza considerando el *Worst Case Execution Time*. Este T_C se establece como el caso más desfavorable entre todos los posibles, como puede ser tener distintos tiempos de proceso en función del procesador que realiza la tarea.

La mejora del *algoritmo myopic* respecto de su versión 'original' [4] está en la cantidad de tareas consideradas cada vez que planifica. El *algoritmo myopic* sólo intenta planificar las *k* tareas más urgentes del conjunto de tareas pendientes. Esto significa una mejora significativa en el tiempo de cálculo necesario, al ser de O(n) frente a $O(n^2)$ de su versión original.

El *algoritmo Myopic* actuaría siguiendo el esquema siguiente, figura 4.4:

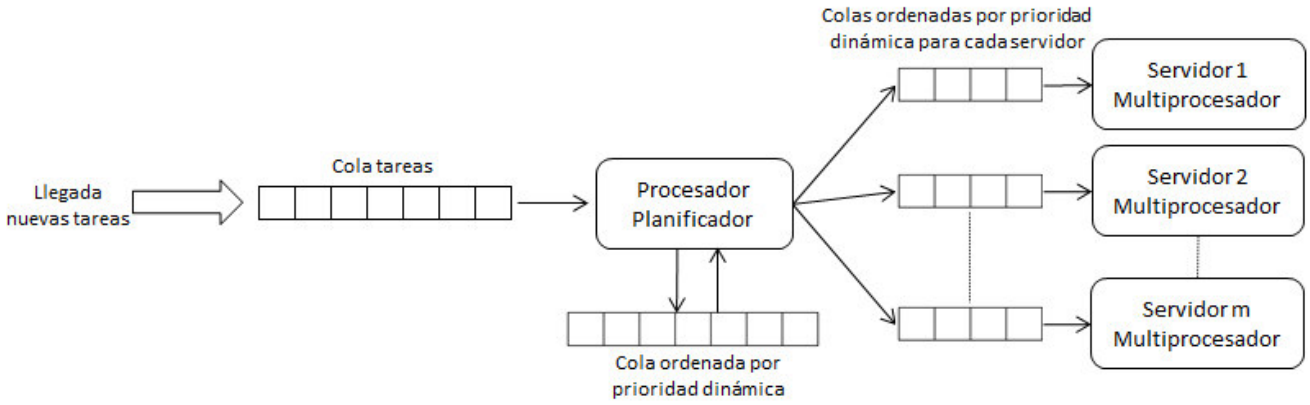


Figura 4.4

En un sistema con distribución de llegadas desconocida (es nuestro caso), tiene mucho sentido considerar la planificación de sólo una cantidad determinada de tareas, puesto las llegadas posteriores, al no ser conocidas ni previsibles, pueden o van a modificar las planificaciones fácilmente. Idealmente, ese número de tareas a planificar debería ser tan bajo como sea posible, pero garantizando también que todos los procesadores tengan carga en todo momento.

Por la tipología de nuestro planteamiento en este TFM, **los algoritmos candidatos para nuestro estudio** serán algoritmos de planificación global, mediante priorización EDF y priorización LLF. Tendremos que considerar igualmente la aplicación del *algoritmo myopic*. Con toda seguridad, deberemos reinterpretar el parámetro T_C, puesto que en nuestro caso no conocemos el tiempo de proceso máximo ni su posible distribución. Además, tampoco se considerará necesario verificar que la planificación sea '*Strongly-feasible*' por el mismo motivo que la reinterpretación necesaria de T_C. Lo comprobaremos en el apartado 7 de la presente memoria.

5. ESPECIFICACIÓN DEL PROBLEMA
(Alcance y Requerimientos del TFM)

El tipo de problema utilizado en el presente estudio, y por lo tanto, **el alcance** de este TFM, es de aplicación muy habitual en entornos de sistemas de tiempo real de varios servidores y varios procesadores, con tasa muy alta de llegadas de solicitudes de trabajo. Ante las solicitudes de trabajo (con tasa de llegadas muy alta), el sistema debe ser capaz de suministrar el resultado final antes de un determinado plazo, minimizando los trabajos entregados fuera de plazo. En este modelo de problema, no se conoce de antemano la duración de los trabajos ni la distribución de las llegadas en ningún momento.

Tomando como referencia las clasificaciones presentadas en el apartado 4 de esta memoria, encuadramos nuestro problema dentro de los sistemas **Soft Real-Time**, de **tareas aperiódicas, no expulsoras y multiprocesador**, además de **multiservidor**, con **distribuciones de llegadas** de solicitudes **estocásticas**.

En el caso que nos ocupa, al tratarse de un estudio con objetivo académico, supondremos que, a nivel de llegadas de solicitudes y tipos de trabajo, *todos los días son iguales*. Aunque sería sencillo, con el software utilizado ARENA, aumentar el estudio y sus resultados a períodos superiores.

Para implementar la parte que corresponde a los **requerimientos de este TFM**, disponemos del modelo de sistema en tiempo real planteado bajo ARENA, que describiremos con más detalle en el apartado 6 de esta memoria. Este modelo está dimensionado de forma tal que pueda ser utilizado con *licencia académica*. Esto nos obligará a escalar la tasa de llegadas, así como a reducir el número de servidores y procesadores para generar colas de espera en el modelo y poder aplicar entonces los algoritmos de secuenciación.

Las llegadas de solicitudes de trabajo a este modelo están basadas en distribuciones calculadas con datos reales recogidos por el departamento. En nuestro modelo, los servidores, encargados de realizar los trabajos, serán 2, con 7 procesadores cada uno. No supondremos que estos servidores son iguales, por lo que, puede ocurrir que un mismo trabajo se ejecute con duraciones diferentes en cada servidor (entorno heterogéneo).

El modelo de ARENA nos permitirá incorporar, vía código Visual Basic, los algoritmos necesarios, tanto para cálculo de la secuenciación en función del método que se elija, como para mantenimiento y tratamiento estadístico de los datos [5]. Estas estadísticas serán las que nos permitirán compensar el desconocimiento de la duración de los trabajos que son solicitados a nuestro sistema de tiempo real.

Además de la selección de algoritmos de secuenciación y su programación bajo VBA, es también requisito de este proyecto la selección de los indicadores que nos permitirán evaluar los resultados extraídos de cada simulación.

Podremos incorporar los módulos y bloques de VBA necesarios para extraer los consiguientes resultados las simulaciones de los diferentes algoritmos y realizar los análisis y comparativas de datos en función de los algoritmos de secuenciación elegidos.

6. DESCRIPCIÓN DEL MODELO DE SIMULACIÓN

El modelo de simulación disponible para la implementación de este TFM está desarrollado con la aplicación ARENA. La aplicación ARENA nos permitirá realizar la simulación de su comportamiento una vez hayamos programado los algoritmos seleccionados dentro de los bloques VBA disponibles para ello.

El modelo consta de 4 estructuras o partes diferenciadas. Con esta organización, seremos capaces de simular con los algoritmos seleccionados, tanto algoritmos de planificación ***global*** como de planificación ***distribuida o local***. Estas partes diferenciadas son:

1. *Generación de las solicitudes de trabajos y definición de los parámetros de tipo de trabajo y duración real.*
2. *Entrada en cola principal y algoritmo de decisión para asignar los trabajos a un servidor determinado*
3. *Entrada en cola de servidor y decisión en caso necesario para asignar un trabajo de esa cola a un servidor con procesador libre de carga de cálculo.*
4. *Ejecución del trabajo en el procesador asignado y toma y registro de datos al finalizar la ejecución.*

Además existen 2 estructuras adicionales para poder implementar bloques adicionales con código Visual Basic en caso de que su uso sea necesario.

A continuación, se muestra una imagen dónde se pueden ver estas **4** estructuras principales comentadas junto con los 2 procesos adicionales.

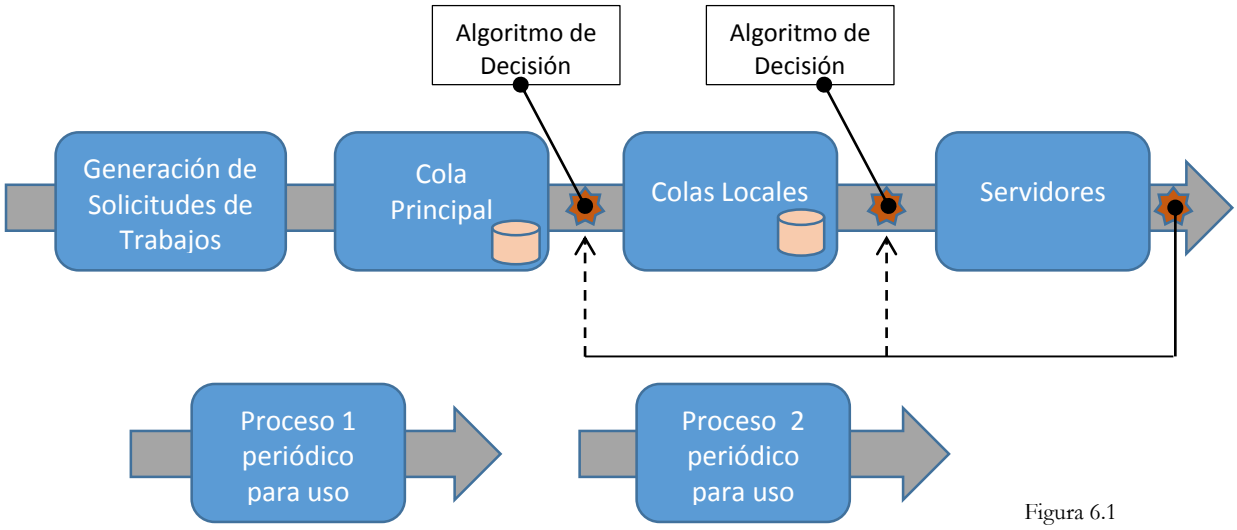
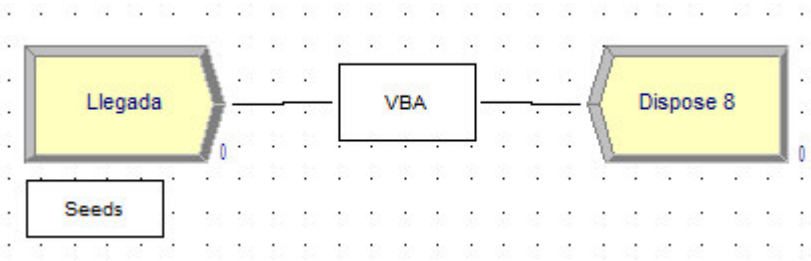


Figura 6.1

Describiremos a continuación con más detalle, las 4 estructuras principales.

- *Bloque 1: Generación de las solicitudes de trabajos y definición de los parámetros de tipo de trabajo y duración real*



Modelo de generación de solicitudes de trabajo

Figura 6.2

Esta generación de solicitudes está basada en los datos recogidos por el departamento en trabajo previo a este estudio de máster. Además de estos datos, contaremos con las distribuciones calculadas para que Arena pueda calcular los tiempos de proceso 'reales' que utilizará durante la simulación. Estos tiempos de proceso 'reales' no serán conocidos para el usuario, tal y como se establece en el enunciado de este caso.

Además de necesitar el tiempo de proceso real, como parámetro interno de Arena, necesitaremos disponer de la distribución de las llegadas para cada tipo de trabajo, en función del tiempo. En este caso, el tiempo vendrá especificado en horas, y la distribución se planteará en forma de histograma. Cada intervalo de 1h de este histograma representa la probabilidad que hay de que se genere una petición de trabajo en ese intervalo. Para disponer de la información completa sobre estos histogramas, se incluyen las tablas para cada tipo de trabajo en el anexo A.

En cada evento generado como llegada por Arena (con tiempo entre llegadas según distribución exponencial), se verifica si corresponde generar una solicitud de trabajo en alguno o varios de los tipos disponibles. Para ello, para cada tipo de trabajo se genera un valor aleatorio según una distribución U[0,1] y se comprueba si su valor es inferior al calculado para el instante de tiempo en cuestión (hora) cuando hemos caracterizado la distribución de llegadas en forma de histograma. En caso positivo, se generará la solicitud de trabajo, con el plazo máximo asignado, y con el

tiempo de proceso real según la distribución disponible de origen, en ambos casos en función del tipo de trabajo. En caso negativo, no se genera solicitud de trabajo de ese tipo.

Esta duración es conocida por Arena para la simulación, pero *no es accesible para nuestro estudio, dado que estamos en un entorno de llegadas aleatorias, con duraciones también aleatorias con distribuciones no conocidas a priori.*

El establecimiento de los plazos de entrega en función del tipo de trabajo queda como sigue (tabla 6.1):

Tipo de trabajo	Plazo máximo
1	Entrega antes de 20 min a partir de la hora de llegada al sistema
2	Entrega antes de las 7:00 am
3	Entrega antes de las 6:00 am
4	Entrega antes de 3h a partir de la hora de llegada al sistema
5	Entrega antes de las 7:00 am
6	Entrega antes de 20 min a partir de la hora de llegada al sistema
7	Entrega antes de 20 min a partir de la hora de llegada al sistema
8	Entrega antes de 3h a partir de la hora de llegada al sistema
9	Entrega antes de 7 días a partir de la hora de llegada al sistema
10	Entrega antes de 24h a partir de la hora de llegada al sistema

Tabla 6.1

Además, aprovecharemos la posibilidad que nos brinda Arena para **fijar** las *series de números aleatorios* (fijando la semilla de generación) que utiliza en todo momento para definir las llegadas de solicitudes de trabajo. Esto nos permite tener mayor consistencia en nuestras evaluaciones de resultados y conclusiones, puesto que en todos los casos simulados, las llegadas al sistema se mantienen iguales y podremos evitar cualquier influencia en los resultados en el caso de que fueran diferentes entre sí. Eliminamos de esta manera la influencia en los resultados por tener llegadas diferentes.

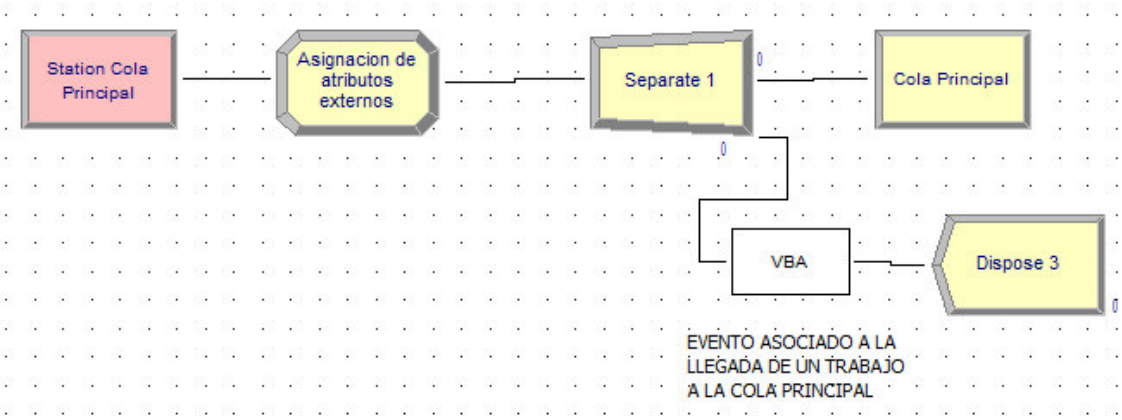
Esta técnica es un método de reducción de la varianza llamado muestreo correlado o números aleatorios comunes. El objetivo de esta técnica es comparar dos o más configuraciones alternativas para el sistema bajo condiciones de experimentación similares. Debemos entender por condiciones de experimentación

similares, los instantes de llegada de las solicitudes de trabajo, y por configuraciones, los distintos algoritmos de secuenciación simulados.

Además de haber utilizado esta técnica en las llegadas de las solicitudes de trabajo, también se utilizan de forma sistemática en los momentos de fijación de tiempos de proceso a cada tarea.

Todo ello nos permite reducir al mínimo la variabilidad en las condiciones de entrada al sistema, y estar, por lo tanto, en disposición de comparar los distintos de métodos de secuenciación bajo condiciones similares.

- *Bloque 2: Entrada en cola principal y algoritmo de decisión para asignar los trabajos a un servidor determinado*



Modelo de cola principal

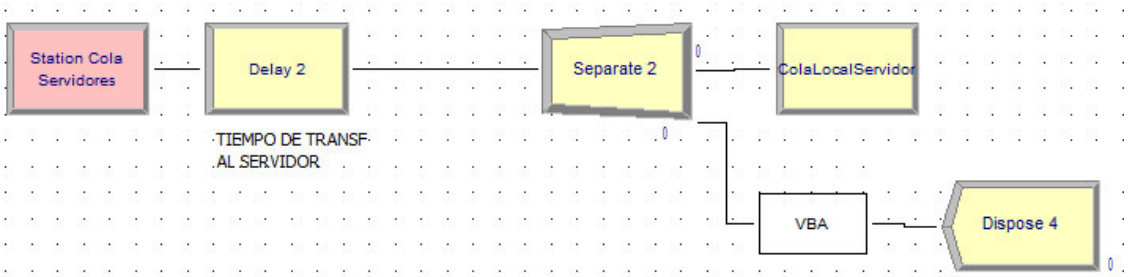
Figura 6.3

En esta estructura, se produce la admisión en cola principal de la solicitud de trabajo. Al entrar en el bloque, la solicitud queda en cola, hasta que el bloque VBA decide que debe trasladarse esta solicitud a los servidores. El bloque VBA llevará la programación de los algoritmos seleccionados para gestionar la cola de solicitudes y decidir cuándo, qué solicitud debe enviar al bloque de cola de servidores, y a cuál de los servidores queda asignado el trabajo solicitado, en función del tipo de algoritmo o gestión seleccionada (EDF, LLT o Myopic en el caso que nos ocupa).

Antes de realizar el envío de la solicitud, **el programa verificará**, con los datos disponibles, **si estima que la solicitud será entregada finalmente dentro de plazo o no al cliente**. En caso negativo, se expulsará la solicitud de la cola y por lo tanto del sistema, antes de consumir CPU sin garantías suficientes para su entrega

dentro de plazo. Esta acción nos permitirá ser más eficientes en la gestión de colas de los servidores, aun entendiendo que la decisión de expulsión es tomada en base a estimaciones disponibles, dado que la carga de procesador a organizar será menor, y por lo tanto, las posibilidades de contar con un sistema planificable, mayores.

- *Bloque 3: Entrada en cola de servidor y decisión en caso necesario para asignar un trabajo de esa cola a un servidor con núcleo libre de carga de cálculo*



Modelo de colas locales

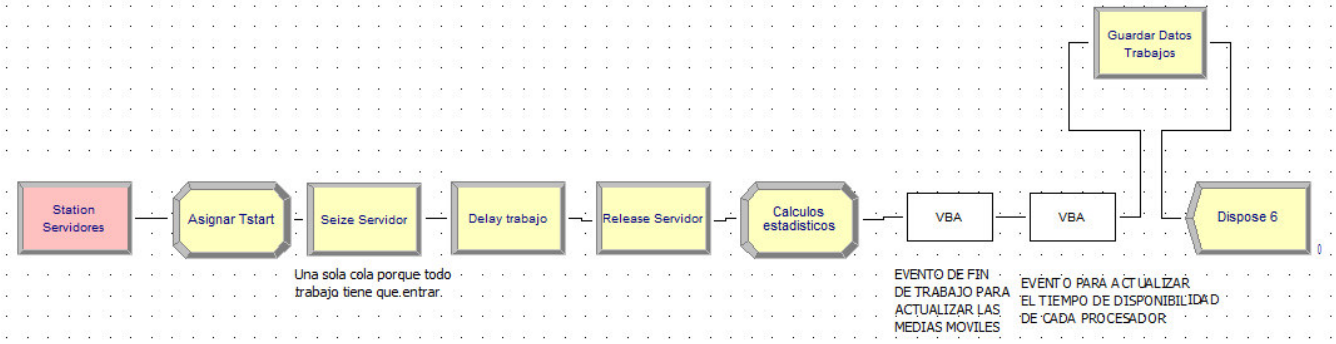
Figura 6.4

En esta estructura, las solicitudes entran sabiendo en cuál de los servidores van a ser atendidas y ejecutadas. A llegar a este bloque, se almacenarán en la cola que corresponda al servidor asignado, y en función del algoritmo seleccionado previamente, será atendida por la estructura siguiente (bloque 4).

Para los casos de algoritmos EDF y LLT, la gestión de colas es simple, puesto que la solicitud es atendida de inmediato por el servidor. Esto es debido a que en ambos casos, la decisión producida en el bloque 2 tiene en cuenta la disponibilidad de capacidad en los servidores antes de realizar el envío de la solicitud (corresponde a un caso claro de planificación global). La gestión de colas de servidores cobra importancia notable para el caso de algoritmo tipo Myopic, cómo comprobaremos en el apartado 7 de esta memoria.

En este bloque, también **se verificará si la solicitud será entregada finalmente dentro de plazo o no al cliente**. En caso negativo, se sacará la solicitud de la cola y se expulsará del sistema antes de consumir CPU sin garantías suficientes para su entrega dentro de plazo. Al igual que en el bloque anterior, la disminución de carga de trabajo de los servidores, nos ayudará a mejorar las posibilidades de éxito de trabajos futuros.

- *Bloque 4: Ejecución del trabajo en el procesador asignado y toma y registro de datos al finalizar la ejecución*



Modelo servidores

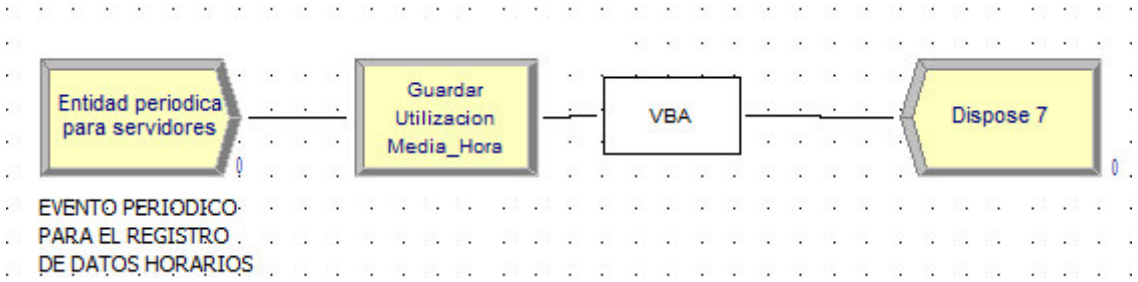
Figura 6.5

En esta etapa, figura 6.5, sólo se realiza la ejecución del servicio, en el servidor asignado en los pasos previos. Al realizar la captura del recurso con el bloque 'Seize', nos permite disponer de una medida de control adicional para confirmar que los algoritmos funcionan correctamente. En caso de funcionamiento erróneo, se generará cola en dicho bloque 'Seize'. Un correcto funcionamiento de los algoritmos no debe generar colas en ese lugar en ningún caso. Tras la ejecución del servicio, se dispone de dos bloques VBA, que nos permitirán, en el primero de ellos, actualizar el control estadístico; en este caso, los datos que se controlan son:

a. Media de la duración de tareas por tipo de trabajo y servidor.
b. Conteo de la cantidad de trabajos que son entregados dentro del plazo máximo permitido.
c. Conteo de la cantidad de trabajos que son entregados fuera del plazo máximo permitido (tarde).
d. Media de adelanto en las entregas (también llamado Earliness).
e. Media de retraso en las entregas (también llamado Lateness).
f. Utilización horaria de cada servidor.

En el segundo de los bloques VBA, se cierra el bucle y se envía el programa a los procedimientos que corresponde, en función del algoritmo de secuenciación utilizado.

Para finalizar esta etapa, se graban en un fichero llamado '*Datos_Tareas'* todos los datos que han caracterizado cada trabajo desde su entrada en el sistema hasta su salida. Estos datos nos permiten hacer un rastreo de la evolución de la simulación y confirmar que la ejecución es correcta, facilitando la tarea de eliminación de errores de código.



Modelo de registro de datos horarios

Figura 6.6

Además, utilizaremos una de las estructuras adicionales (figura 6.6) comentadas anteriormente para controlar y guardar en archivo los datos siguientes, en cada hora de simulación, y en función del servidor y tipo de trabajo:

a. Ocupación de cada servidor
b. Cantidad de trabajos entregados en plazo
c. Cantidad de trabajos entregados fuera de plazo
d. Cantidad de trabajos no iniciados por estimar que se entregarán en plazo
e. Media de adelanto en las entregas (también llamado Earliness).
f. Media de retraso en las entregas (también llamado Lateness).
g. Media de la duración de trabajos.

Estos datos quedan guardados en los archivos *'Ocup_Servidor'* y *'Estadisticas_Hora'*

Una vez finaliza la simulación (de 24h en nuestro caso), se recuperarán todos los datos guardados para consolidar la evaluación y los resultados del presente trabajo.

7. PLANTEAMIENTO Y DECISIÓN SOBRE PROPUESTAS ALTERNATIVAS

Nuestro objetivo consiste en planificar o establecer el secuenciamiento en entornos heterogéneos, con el objetivo principal de cumplir los plazos de entrega máximos establecidos. Estos plazos máximos dependerán del tipo de tarea, existiendo 10 tipos de tareas definidos en nuestro sistema. Las tareas tienen unos instantes de llegada desconocidos (patrón de llegadas no conocido). Además tampoco se dispone del tiempo de ejecución necesario para cada tarea. Al ser un entorno heterogéneo, una tarea podrá tener tiempos de ejecución diferentes dependiendo del servidor en el que se ejecute finalmente.

El hecho de no conocer los tiempos de proceso de las tareas a su llegada al sistema nos impedirá utilizar métodos de planificación estática, puesto que no se dispondrá en ningún momento de toda la información necesaria, como puede ser el tiempo de proceso. Su uso con aproximaciones nos facilitaría resultados poco robustos. Teniendo en cuenta que las tareas tienen diferentes plazos máximos de entrega, la planificación estática será poco flexible y podrá planificar tareas menos urgentes antes que las más urgentes, con el riesgo directo de no cumplir plazos con estas últimas. Este tipo de planificación sería poco flexible para nuestro caso.

Ante la incertidumbre sobre las llegadas de tareas futuras y de sus tiempos de proceso, la opción de planificación online es la más adecuada cuando no conocemos esta distribución de las llegadas. Por el contrario, este tipo de planificación cuenta con el conocimiento de la duración de las tareas, característica de la que no disponemos. Para solventarlo utilizaremos la esperanza de tiempo de proceso para cada tarea, en función del histórico de tiempos de proceso disponible [5]. Este histórico se registrará a medida que las tareas vayan siendo ejecutadas y finalizadas. El trabajo con esperanzas de tiempo de proceso corresponde, en el sentido estricto a un tipo de planificación llamado estocástico [6].

A continuación se exponen los algoritmos seleccionados para el presente estudio comparativo.

7.1 Selección de alternativas

Los métodos seleccionados, teniendo en cuenta lo comentado más arriba, son los siguientes:

- Earliest Deadline First (EDF)

Es el método por excelencia para esta tipología de problemas en que queremos que la entrega de las tareas procesadas sea realizada antes de un cierto límite de tiempo. Utilizaremos un tipo de Planificación Global, en la que tendremos una cola principal a la que llegarán las tareas. Desde esa cola, priorizando por el menor de los plazos de entrega máximo de cada tarea, enviaremos trabajo a los servidores a medida que los procesadores vayan quedando libres.

- Least Laxity First (LLF)

Este método también es una referencia para este tipo de problemas. Utilizaremos el mismo planteamiento de Planificación Global con cola de tareas que en el caso anterior. La prioridad o posición en la cola se determinará en función de la holgura que tenga cada tarea. Recordamos la definición de holgura:

$$Holgura = Plazo\ Entrega\ Máximo - Tiempo\ de\ Ejecución$$

- Myopic Algorithm

Este método es más desarrollado que los dos anteriores tal y como hemos introducido en el punto anterior. Pese a utilizar el planteamiento de Planificación Global, se gestionarán también las colas de acceso a cada uno de los servidores, puesto que el algoritmo asigna la tarea siempre y cuando la planificación en el servidor seleccionado sea viable una vez haya incluido esa nueva tarea. La bibliografía [4] indica que se trata de un método heurístico con muy buenos resultados.

En este algoritmo, la prioridad en la cola principal será en función de la fecha de entrega máxima de cada tarea, como el método EDF. Las colas de cada servidor serán gestionadas desde el algoritmo Myopic, que es quién decidirá el orden de ejecución de las tareas para mantener viable la planificación y respetar los plazos de entrega máximos. En estas colas locales, se utilizará la priorización por plazo de entrega máximo. La tarea con menor plazo de entrega máximo será la más prioritaria. Estableceremos una ventana de planificación variable, de forma a mantener un

máximo de 7 elementos en cola para cada servidor. De esta forma garantizaremos 2 aspectos:

1. Tendremos 7 tareas en cola, en espera de ejecución. En caso de quedar libres los 7 procesadores, éstas serían atendidas mientras el algoritmo planifica el lote siguiente. No habría un tiempo ocioso en el servidor operando de esta manera, salvo que no quedasen solicitudes de trabajo en Cola Principal
2. La entrada en cola de servidor se realiza lo más tarde posible, así podremos priorizar con el mayor número de solicitudes de trabajo posible en Cola Principal en cada momento, evitando planificar tareas demasiado pronto. Esto es importante, puesto que nuestro sistema es no-expulsivo.

En lo relativo al parámetro T_C mencionado en el apartado 4, deberemos tomar en su lugar el *tiempo de proceso esperado*, en función de los históricos de tiempos de proceso que tengamos en cada momento. Al ser tareas de duraciones no definidas y estocásticas, no tiene sentido pensar que T_C debe ser el peor de los tiempos de proceso que tengamos hasta el momento para un tipo de tarea. Estaríamos haciendo un planteamiento demasiado conservador, y a la larga, no adecuado al uso que se persigue. El algoritmo programado nos permitirá utilizar T_C adecuado para cada servidor concreto. Recordemos que nuestro entorno considerado es heterogéneo.

Esta metodología exige una comprobación para confirmar que las propuestas de planificación son '*Strong-feasible*'. En nuestro caso, no vemos necesario este aspecto, puesto que estaríamos haciendo una comprobación con unos tiempos de proceso aproximados. El resultado podría ser irreal, puesto los tiempos de proceso son calculados en base a un histórico disponible en cada momento, y podría hacernos descartar planificaciones correctas sólo por el hecho de no tener unas duraciones lo suficientemente robustas. Por otro lado, al no conocer las llegadas de solicitudes de trabajo futuras, podríamos estar verificando '*Strong-feasibility*' con un conjunto de tareas que deberá cambiar con el tiempo. No se nos asegura que el conjunto de tareas sea el correcto para realizar esa comprobación. En nuestro caso, tenemos que obviar esta comprobación y verificar que la tarea que incluimos en planificación es entregable dentro del plazo exigido y que la planificación del servidor seleccionado para nuestra tarea entregará cada tarea dentro de su plazo con la nueva planificación.

7.2 Selección de indicadores para evaluación de resultados

Para poder evaluar y comparar las prestaciones de los tres algoritmos seleccionados anteriormente, necesitaremos seleccionar una serie de indicadores,

teniendo en cuenta que el objetivo es el de respetar en todos los casos posibles los plazos de entrega máximos fijados.

Para ello, utilizaremos los indicadores siguientes:

Número de tareas entregadas a tiempo: Es el número de tareas entregadas antes del plazo de entrega máximo prefijado al llegar al sistema.

Número de tareas entregadas tarde: Es el número de tareas entregadas fuera de plazo de entrega máximo.

Número de tareas expulsadas: Es el número de tareas expulsadas del sistema, bien sea desde la cola principal, o bien desde las colas de los servidores. Es consecuencia de la verificación que se realiza antes de enviar la tarea a la cola del servidor o al servidor para su ejecución. Esta verificación se realiza para evitar consumir tiempo de CPU con una tarea que 'intuimos' puede llegar fuera de plazo en su entrega. En una planificación con tiempos de procesos conocidos, este parámetro tendría relevancia sólo en el momento de enviar la tarea desde la cola principal hasta los servidores, puesto que las planificaciones en cada servidor tienen en cuenta esta duración y los plazos de entrega se respetarán. En nuestro caso, al estar trabajando con duraciones de tareas estimadas (con la esperanza de tiempo de procesos), no sería suficiente la verificación antes de enviar la tarea a la cola del servidor. Debemos añadir el control también antes de enviar la tarea para su ejecución, puesto que en ese momento evitamos toda influencia de tareas anteriores, puesto que ya han sido ejecutadas y sus tiempos de procesos son, a partir de ese momento, conocidos.

Earliness: Media de adelanto en la entrega de las tareas servidas dentro del plazo asignado.

Lateness: Media de retraso en la entrega de las tareas servidas fuera del plazo asignado.

Ocupación de servidor por hora: Lo utilizaremos para confirmar el aprovechamiento de los recursos y confirmar si están saturados. La tipología del problema nos obliga a registrar la ocupación horaria, y no diaria, debido a la gran diversidad de tipos de tareas, así como a la nula previsibilidad en sus llegadas al sistema. La fórmula que aplicaremos para conocer la utilización horaria de cada servidor es la siguiente:

$$Utilización\ Servidor = \frac{\sum_t C_t}{60}$$

Dónde C_t es el tiempo de proceso de la tarea t atendida durante la hora evaluada. Se sumarán todos los tiempos de proceso de las tareas atendidas (en minutos) y se

dividirá por 60 minutos para obtener el valor de utilización horaria en tanto por uno para cada servidor.

7.3 Descripción de los algoritmos e implementación

7.3.1 Algoritmo EDF (Earliest Deadline First)

Este algoritmo está basado en la asignación de prioridad dinámica en función del plazo de entrega máximo que deba respetarse para cada tarea. La tarea que tendrá la prioridad máxima será aquella que tenga el menor plazo máximo de entrega entre todas las tareas pendientes de acceder al recurso para ser ejecutadas.

Este tipo de algoritmo o gestión de prioridades basadas en el plazo máximo de entrega es óptimo para la configuración que contempla: sistema expulsor, monoprocesador en tiempo real con tareas aperiódicas con instantes de llegada al sistema desconocidos. En esta configuración, el algoritmo EDF es capaz de alcanzar un uso máximo de la capacidad del procesador. No es el caso para sistemas multiprocesador.

La gestión EDF para sistemas multiprocesador (y multiservidor) se realizará mediante Planificación Global [1]. Esto supone que todas las solicitudes de tareas son acumuladas en una única cola (denominada Cola Principal), desde la que, el algoritmo elegirá la tarea con mayor prioridad EDF y la enviará al servidor con mayor disponibilidad que haya en cada momento. Esta configuración es la idónea para el tipo de problema que estamos afrontando, puesto que no conocemos de manera cierta los tiempos de proceso de las trabajos. En diversos puntos del código implementado, como se detallará a medida que se expliquen los algoritmos, es necesario disponer de la duración de las tareas para poder predecir si seremos capaces de suministrar el resultado de la tarea antes del plazo máximo estipulado. Al tratarse de un sistema con llegadas no predecibles de tareas con duraciones desconocidas, deberemos utilizar el histórico de duraciones que nuestro código irá registrando a medida que se vayan ejecutando las tareas en cada servidor [5]. El criterio establecido es disponer de al menos 15 datos para empezar a ser utilizados como duraciones medias 'fiables'. Mientras no se cumpla disponer de esos 15 datos, las solicitudes de trabajo se entenderán como 'entregables dentro del plazo máximo'. Es comprensible, dado que sin ejecución de trabajos, nos será imposible disponer de datos históricos.

La implementación del algoritmo se plantea, en nuestro caso, bajo 4 bloques principales de funcionamiento, que llamaremos: Generación de Tareas, Gestión de la Cola Principal, Gestión de las Colas locales o de los Servidores y el bloque final que incluye la ejecución de las tareas en servidores junto con los registros estadísticos de seguimiento. Detallando cada uno de estos bloques:

Generación de Tareas: Es el bloque suministrado por el departamento de Sistemas, Automática e Informática Industrial (ESAII). Es el responsable de generar las solicitudes de trabajo conforme a las distribuciones de llegadas previamente calculadas en función de los datos reales recogidos por muestreo a pie de campo. Cada una de estas solicitudes pasa directamente al bloque de Gestión de la Cola Principal. La generación de cada llegada al sistema se realiza mediante una distribución exponencial de media 0.07min, fijando la semilla de generación de número seudoaleatorios. Esto último nos permite obtener comparativas más fiables entre los 3 métodos probados, dado que reducimos la varianza del factor aleatorio en los instantes de llegada.

Gestión de la Cola Principal: Esta parte de código se ejecuta cada vez que llega una solicitud de trabajo nueva al sistema y cada vez que queda libre algún procesador. Como vemos en el diagrama de flujo de la figura 7.1, se verifica cuántos procesadores hay disponibles, y en caso positivo se envían tareas mientras haya solicitudes en cola y ***mientras queden procesadores disponibles***, priorizando el servidor que tenga más procesadores disponibles en cada momento. Justo antes de enviar la tarea al servidor correspondiente se elimina la petición de la cola principal y se verifica que será posible suministrar el resultado de la tarea antes del plazo máximo. En caso de que no lo sea, la tarea no se enviará al servidor, y se contabilizará como 'Tarea Expulsada del Sistema desde Cola Principal'.

Uno de los puntos fuertes de este código VBA es que cada vez que accedemos a él, se verifica cuántos procesadores hay disponibles y se les envían tantas tareas como disponibilidad tienen (siempre que hayan suficientes en cola principal) de una sola vez. No se realiza un único envío de tarea en cada acceso. Nos permite optimizar recursos en la asignación de servidores a las tareas en cola.

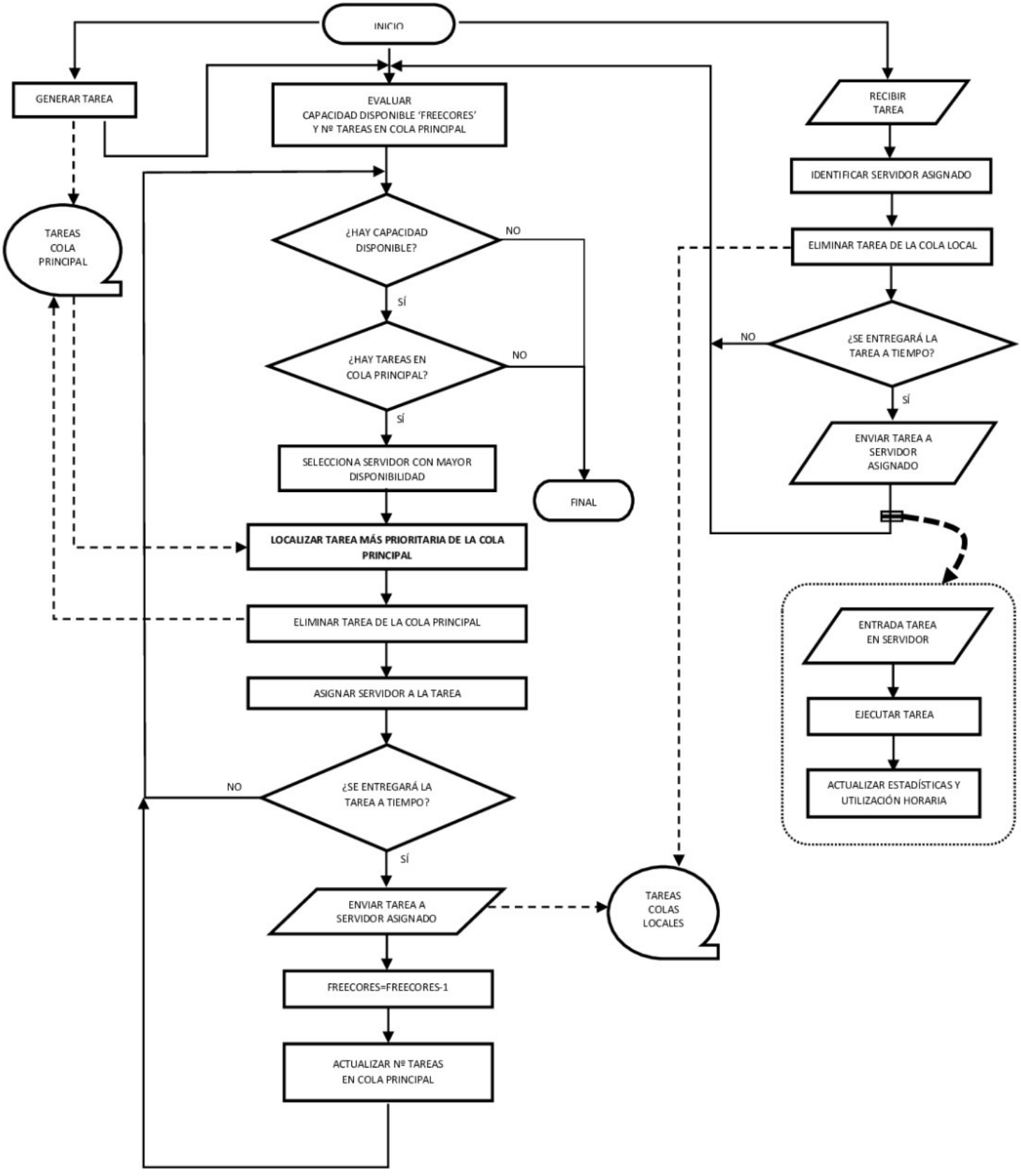

INICIO
GENERAR TAREA
EVALUAR
CAPACIDAD DISPONIBLE 'FREECORES'
Y Nº TAREAS EN COLA PRINCIPAL
RECIBIR
TAREA
IDENTIFICAR SERVIDOR ASIGNADO
TAREAS
COLA
PRINCIPAL
¿HAY CAPACIDAD
DISPONIBLE?
NO
SÍ
ELIMINAR TAREA DE LA COLA LOCAL
NO
¿SE ENTREGARÁ LA
TAREA A TIEMPO?
SÍ
¿HAY TAREAS EN
COLA PRINCIPAL?
NO
SÍ
ENVIAR TAREA A
SERVIDOR
ASIGNADO
SELECCIONA SERVIDOR CON MAYOR
DISPONIBILIDAD
FINAL
LOCALIZAR TAREA MÁS PRIORITARIA DE LA COLA
PRINCIPAL
ELIMINAR TAREA DE LA COLA PRINCIPAL
ENTRADA TAREA
EN SERVIDOR
ASIGNAR SERVIDOR A LA TAREA
EJECUTAR TAREA
ACTUALIZAR ESTADÍSTICAS Y
UTILIZACIÓN HORARIA
NO
¿SE ENTREGARÁ LA
TAREA A TIEMPO?
SÍ
ENVIAR TAREA A
SERVIDOR ASIGNADO
TAREAS
COLAS
LOCALES
FREECORES=FREECORES-1
ACTUALIZAR Nº TAREAS
EN COLA PRINCIPAL

Figura 7.1

Gestión de las Colas Locales: Este bloque se ejecuta cada vez que llega una tarea a la cola de un servidor. Se elimina la tarea de la Cola Local en la que figura y se realiza una verificación adicional para comprobar que será posible suministrar el resultado de la tarea antes del plazo máximo. En caso negativo, se expulsa la tarea y quedará contabilizada como 'Tarea Expulsada del Sistema desde Cola de Servidor *i*', siendo *i* el número del servidor que tenía asignado para su ejecución. Al tratar de un tipo de Planificación Global y por el modelado aplicado, el tamaño de las colas de los servidores será 1 en todo momento.

Ejecución de Tareas: Este bloque tan sólo realiza la ejecución de la tarea en el servidor asignado previamente. Su función principal, a nivel de codificación VBA es el registro de los datos estadísticos que se utilizan durante la simulación. Entre estos datos figuran: Número de tareas entregadas a tiempo, número de tareas entregadas tarde, la media de Lateness, media de Earliness y la ocupación de los servidores por hora. Otro dato que se registra y que es de una importancia máxima es la duración de las tareas. Este dato se recoge en su valor medio histórico, distinguiendo tipo de trabajo y servidor en el que se ejecuta el trabajo. Es un dato crucial para poder establecer si se debe expulsar una solicitud de trabajo antes de iniciar su ejecución, como hemos visto en bloques anteriores.

7.3.2 Algoritmo LLF (Least Laxity First)

Este algoritmo también es conocido como *Least Slack Time*. Al igual que el algoritmo EDF, se trata de un algoritmo basado en la gestión dinámica de las prioridades de cada tarea. En este caso, la prioridad se asigna en función de la 'holgura'. La 'holgura', como hemos descrito anteriormente, se define como:

$$Holgura = Plazo\ Entrega\ Máximo - Tiempo\ de\ Ejecución$$

Aquella tarea cuya holgura sea menor que todas las de las demás, será la tarea con mayor prioridad.

Al igual que el algoritmo EDF, el algoritmo LLF es óptimo dentro de los mismos supuestos; no siéndolo tampoco para el caso que nos ocupa de sistema multiprocesador.

La gestión LLF para sistemas multiprocesador (y multiservidor) se realizará mediante Planificación Global [1], de forma absolutamente análoga al diagrama de

flujo para el algoritmo EDF (figura 7.1), con las mismas motivaciones. El único cambio entre ambos algoritmos, obviando el de los resultados de la simulación, es el criterio de priorización utilizado en el momento de seleccionar las tareas desde la Cola Principal. En este caso, evidentemente, se utilizará una priorización LLF.

Adicionalmente, serán prioritarias las tareas con tiempos de proceso estimados nulos, es decir aquellos de los que no se posean registros de duración[5]. Esto es necesario para nutrir los registros estadísticos de duraciones de proceso para poder alimentar correctamente los cálculos de holgura. Si no fuera así, la holgura en estos casos equivaldría al plazo máximo permitido, por lo que sería posible que el trabajo se ejecutara demasiado tarde.

La distribución de los bloques y sus funcionalidades son exactamente iguales que para el caso de priorización EDF.

7.3.3 Algoritmo MYOPIC

Es un algoritmo para multiprocesador, utilizando métodos heurísticos para la asignación de servidor a las solicitudes de trabajos que figuren en la Cola Principal. Este procedimiento transforma de alguna manera, un problema multiservidor en *i* problemas monoservidor, dónde *i* representa el número de servidores de nuestro problema.

En nuestro problema, además de disponer de varios servidores, debemos tener en cuenta que cada servidor es multiprocesador. El método Myopic [4] utiliza técnicas heurísticas para la asignación de tareas a los servidores. Cada tarea que se intenta planificar es asignada al primer servidor que pueda ofrecer una planificación viable. Entiéndase planificación viable como aquella que hace que todas las tareas consideradas son entregadas cumpliendo el plazo máximo fijado.

En nuestro marco de trabajo, debemos destacar:

1. Las tareas son independientes, aperiódicas, no-exclusivas y pueden procesarse en cualquiera de los servidores.

2. No disponemos de antemano de las duraciones de los trabajos, por lo que todo cálculo que necesite de las duraciones de los trabajos será realizado utilizando los datos históricos que el programa acumula durante su ejecución [5]. Esto dificulta la confirmación de planificación viable cuando se realiza la asignación de tareas a servidores.

3. La *ventana de planificación* es definida '***dinámica'***, dado que, por la naturaleza de las llegadas y de las duraciones desconocidas, nos será más favorable (ver apdo 7.1) poder limitar el número de tareas máximo en cola de cada servidor. Esta decisión nos permite gestionar de manera más robusta la prioridad de las tareas en Cola Principal, puesto que esperamos un tiempo casi máximo hasta realizar la planificación hacia las colas de los servidores. Este tiempo es máximo si la planificación se realiza mediante el algoritmo EDF o LLF anteriores.

La implementación del algoritmo (figura 7.2) para este método se plantea bajo 5 bloques principales de funcionamiento, que llamaremos: Generación de Tareas, Gestión de la Cola Principal Myopic, Gestión de las Colas locales Myopic o de los Servidores, Cálculo de Planificación para una tarea y el bloque final que incluye la ejecución de las tareas en servidores junto con los registros estadísticos de seguimiento. Para este algoritmo, ha sido necesario establecer una estructura de matrices de colas en paralelo con la gestión de colas que nos ofrece *Arena*, para poder comprobar correctamente las viabilidades de las sucesivas planificaciones. Los componentes de estas matrices están ordenados en función de la prioridad EDF. El objeto de estas matrices es disponer de una 'foto fija' de cada cola en los momentos de cálculo, frente a la gestión dinámica que tenemos para las colas de *Arena*.

Detallando cada uno de estos bloques:

Generación de Tareas: Es el bloque suministrado por ESAII, igual que para los métodos EDF o LLF. Es el responsable de generar las solicitudes de trabajo conforme a las distribuciones de llegadas previamente calculadas en función de los datos reales recogidos por muestreo a pie de campo. Cada una de estas solicitudes pasa directamente al bloque de Gestión de la Cola Principal. La generación de cada llegada al sistema se realiza mediante una distribución exponencial de media 0.07min, fijando la semilla de generación de número seudoaleatorios. Esto último nos permite obtener comparativas más fiables entre los 3 métodos probados, dado que retiramos la influencia del factor aleatorio en los instantes de llegada.

Gestión de la Cola Principal Myopic: Esta parte de código se ejecuta cada vez que llega una solicitud de trabajo nueva al sistema o cada vez que finaliza la ejecución de algún trabajo en los servidores (es decir, cuando queda libre algún procesador). Como vemos en el diagrama de flujo de la figura 7.2, se verifica cuántas tareas podemos ubicar en las colas de los servidores (dato recogido bajo la variable '*forecast*'), en función de los huecos que haya en cada cola y en función de los procesadores ocupados en ese momento. La planificación se realiza para ese número de tareas, siempre y cuando tengamos las tareas suficientes en la Cola Principal. En el caso de

que no sea así, la planificación se realiza para las tareas existentes hasta ese momento. La planificación para una tarea se verifica mediante el subproceso '*BuscarServMasRapido'* que se describirá como bloque propio más adelante en esta memoria. Antes de proceder a esa verificación o cálculo, se comprueba si será posible suministrar el resultado de la tarea antes del plazo máximo. En caso positivo, se procede a comprobar si existe planificación en algún servidor para la solicitud de trabajo. Si se encuentra un servidor dónde sea planificable, se procede a eliminar la solicitud de la Cola Principal, se actualizan las matrices de Colas, tanto Principal como del Servidor que se haya seleccionado y se remite la tarea a la Cola del Servidor seleccionado. Si se estima que la entrega tendrá lugar fuera del plazo máximo especificado, no se ejecuta el trabajo, se elimina la solicitud de la Cola Principal, se actualiza la matriz correspondiente a la cola principal y se procede con el intento de planificación de la tarea siguiente, de acuerdo con las prioridades; en este caso, se contabilizará como 'Tarea Expulsada del Sistema desde Cola Principal'.

El punto fuerte de este código VBA es que cada vez que lanzamos un bucle de planificación Myopic, lo realizamos para una cantidad de solicitudes variable, en función de la carga y de las colas de espera de los servidores. Nos permite retardar la asignación de servidor a las tareas un máximo de tiempo, mientras que tenemos carga asegurada en los servidores.

Gestión de las Colas Locales Myopic: Este bloque se ejecuta cada vez que llega una tarea a la cola de un servidor o bien cada vez que se realiza un bucle de planificación Myopic (haya o no haya solicitudes de trabajo en Cola Principal). Esta gestión se realiza para cada uno de los servidores disponibles. Siempre que haya solicitudes de trabajo en cola de servidor, y que haya capacidad disponible (procesadores libres) para ejecución en el mismo servidor, el código enviará, respetando la prioridad de las tareas, trabajos mientras haya procesadores libres y solicitudes de trabajo en cola de servidor. De manera análoga al caso de planificación EDF, antes de enviar el trabajo al servidor, se procederá a su eliminación de la cola del servidor, a la actualización de la matriz de cola de servidor, y a la comprobación de si será posible suministrar el resultado de la tarea antes del plazo máximo. Si se estima que la entrega tendrá lugar fuera del plazo máximo especificado, no se ejecuta el trabajo, y se procede con la tarea siguiente. Esta tarea expulsada se contabilizará como 'Tarea Expulsada del Sistema desde Cola de Servidor *i* ', siendo *i* el número del servidor que tenía asignado para su ejecución.

Para el que el algoritmo sea capaz de comprobar en el bloque de Cálculo de Planificación Myopic si un conjunto de tareas tiene una planificación viable en un servidor con *n* procesadores es necesario tener trazabilidad sobre los trabajos en

curso en cada procesador de cada servidor. Esto se consigue en el código propuesto actualizando la variable '*varDdTareasEnCurso*', donde se almacena el instante de tiempo de finalización previsto para cada trabajo en curso, antes de enviar cada tarea a su servidor asignado y procesador correspondiente. Una vez finalizada la ejecución de un trabajo, la variable '*varDdTareasEnCurso*' debe ser de nuevo actualizada, guardando un valor 0 en la posición del procesador que ha quedado disponible.

Cálculo de Planificación Myopic: Este bloque se ejecuta cada vez que se quiere comprobar si una solicitud de trabajo es viable dentro de una de las planificaciones de servidores. Entraremos al algoritmo (ver figura 7.3) con todos los datos de la solicitud de trabajo y saldremos de él con los mismos datos, además del servidor al que ha sido asignado, en caso de que se haya encontrado una planificación viable. Si no ha encontrado ninguna planificación, el número de servidor asignado será el 0. Si ha encontrado más de un servidor con planificación viable, se entregará el servidor que nos devuelva antes la ejecución del trabajo. Para el cálculo de la planificación, se procede a través de los pasos siguientes:

1. Si algún servidor no tiene cola de trabajos pendientes:
    a. Si hay algún procesador libre, la tarea queda inmediatamente asignada y saldremos del bloque.
    b. Si todos los procesadores están ocupados, se comprueba cuál es el que acabará antes y se verifica si el trabajo que queremos asignar podrá acabar antes del plazo máximo establecido.

   Si se produce asignación de trabajo en este apartado, ya no se verifican los servidores restantes. Se considera un resultado suficientemente bueno para seguir con la simulación.
2. Cuando no hay servidores sin cola de trabajos:
    a. Se verifica si hay algún procesador disponible. Es un caso un poco extraño, dado que hay cola de tareas, pero es una posibilidad que debemos tener en cuenta, por si la tarea que queremos planificar tuviera prioridad sobre la más prioritaria en cola del servidor, y pudiera originar una planificación viable.
    b. Si no hay procesador libre, se procede a buscar la posición hipotética en la cola del servidor, en función de la prioridad para el trabajo que queremos planificar.
3. Tanto si venimos de 2a como si lo hacemos desde 2b, el programa comprobará, que además de poder entregar el trabajo que queremos planificar a tiempo, también podrá entregar a tiempo el resto de tareas que ya figuraban en la cola.

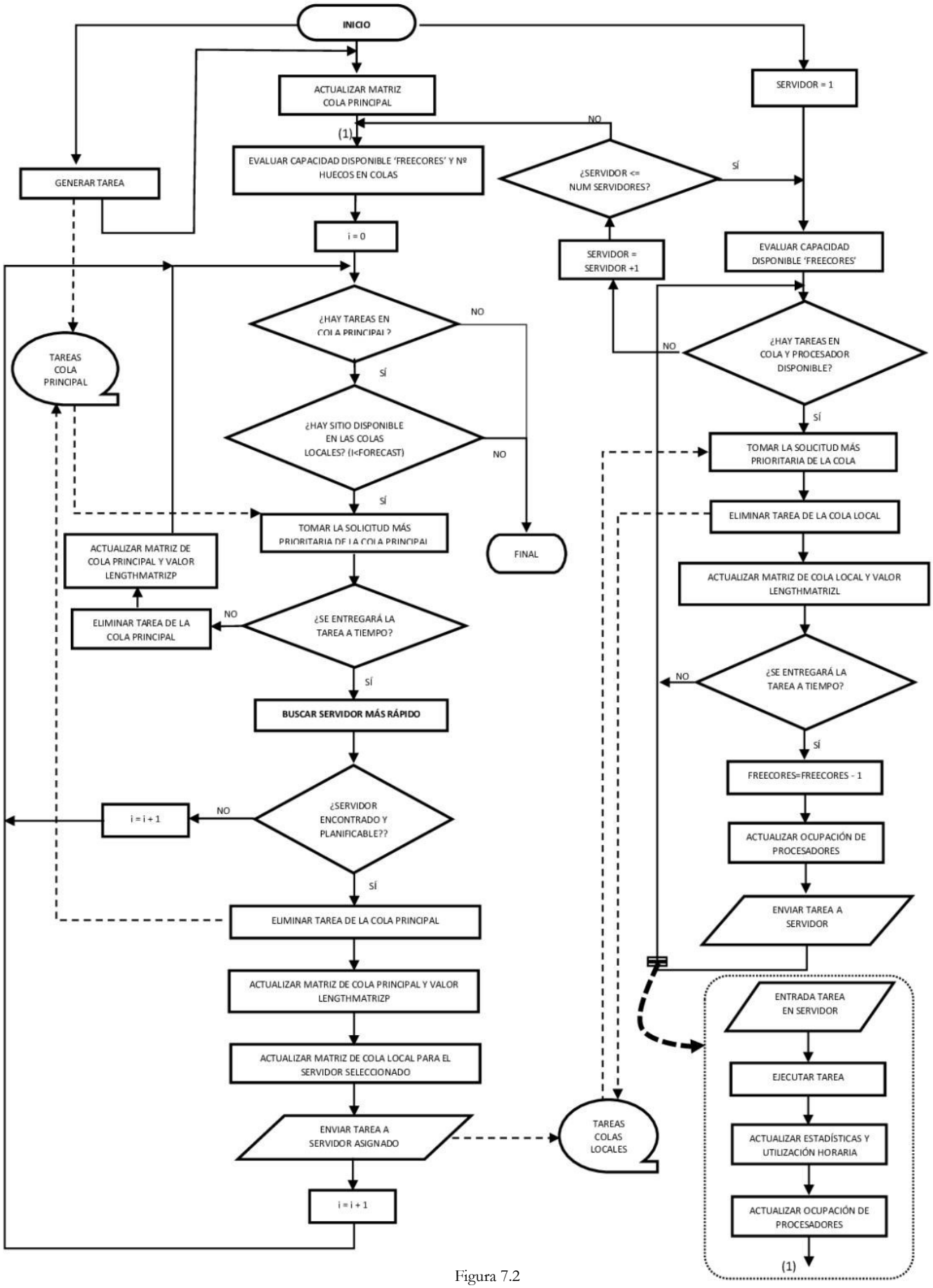

INICIO
ACTUALIZAR MATRIZ COLA PRINCIPAL
(1)
EVALUAR CAPACIDAD DISPONIBLE 'FREECORES' Y Nº HUECOS EN COLAS
GENERAR TAREA
i = 0
¿HAY TAREAS EN COLA PRINCIPAL?
NO
SÍ
TAREAS COLA PRINCIPAL
¿HAY SITIO DISPONIBLE EN LAS COLAS LOCALES? (I<FORECAST)
NO
SÍ
FINAL
TOMAR LA SOLICITUD MÁS PRIORITARIA DE LA COLA PRINCIPAL
ACTUALIZAR MATRIZ DE COLA PRINCIPAL Y VALOR LENGTHMATRIZP
ELIMINAR TAREA DE LA COLA PRINCIPAL
NO
¿SE ENTREGARÁ LA TAREA A TIEMPO?
SÍ
BUSCAR SERVIDOR MÁS RÁPIDO
i = i + 1
NO
¿SERVIDOR ENCONTRADO Y PLANIFICABLE??
SÍ
ELIMINAR TAREA DE LA COLA PRINCIPAL
ACTUALIZAR MATRIZ DE COLA PRINCIPAL Y VALOR LENGTHMATRIZP
ACTUALIZAR MATRIZ DE COLA LOCAL PARA EL SERVIDOR SELECCIONADO
ENVIAR TAREA A SERVIDOR ASIGNADO
i = i + 1
TAREAS COLAS LOCALES
NO
¿SERVIDOR <= NUM SERVIDORES?
SÍ
SERVIDOR = SERVIDOR +1
SERVIDOR = 1
EVALUAR CAPACIDAD DISPONIBLE 'FREECORES'
NO
¿HAY TAREAS EN COLA Y PROCESADOR DISPONIBLE?
SÍ
TOMAR LA SOLICITUD MÁS PRIORITARIA DE LA COLA
ELIMINAR TAREA DE LA COLA LOCAL
ACTUALIZAR MATRIZ DE COLA LOCAL Y VALOR LENGTHMATRIZL
NO
¿SE ENTREGARÁ LA TAREA A TIEMPO?
SÍ
FREECORES=FREECORES - 1
ACTUALIZAR OCUPACIÓN DE PROCESADORES
ENVIAR TAREA A SERVIDOR
ENTRADA TAREA EN SERVIDOR
EJECUTAR TAREA
ACTUALIZAR ESTADÍSTICAS Y UTILIZACIÓN HORARIA
ACTUALIZAR OCUPACIÓN DE PROCESADORES
(1)

Figura 7.2

Es importante destacar que todos estos cálculos necesitan de unos tiempos de proceso de trabajos para poder ser realizados de una manera robusta. En nuestro caso, al no conocer los tiempos de proceso de cada trabajo, debemos realizar los cálculos en base a los datos históricos de tiempos de ejecución [5].

Ejecución de Tareas: Este bloque, al igual que en el algoritmo EDF o LLF, tan sólo realiza la ejecución de la tarea en el servidor asignado previamente. Su función principal, a nivel de codificación VBA es el registro de los datos estadísticos que se utilizan durante la simulación. Entre estos datos figuran: Número de tareas entregadas a tiempo, número de tareas entregadas tarde, la media de Lateness, media de Earliness y la ocupación de los servidores por hora. Otro dato que se registra y que es de una importancia máxima es la duración de las tareas. Este dato se recoge en su valor medio histórico, distinguiendo tipo de trabajo y servidor en el que se ejecuta el trabajo. Es un dato crucial para poder establecer si se debe expulsar una solicitud de trabajo antes de iniciar su ejecución, como hemos visto en bloques anteriores. La actualización de la variable '*varDdTareasEnCurso*' antes mencionada se produce en este bloque, al finalizar cada trabajo. Se guardará un valor 0 en la posición del procesador que ha quedado disponible. Al finalizar la ejecución de un trabajo, se ejecutará la gestión de cola principal, y no la de cola local. El objetivo es poder *tener en cuenta trabajos más urgentes, que hayan podido llegar y no haber sido planificados hasta el momento.*

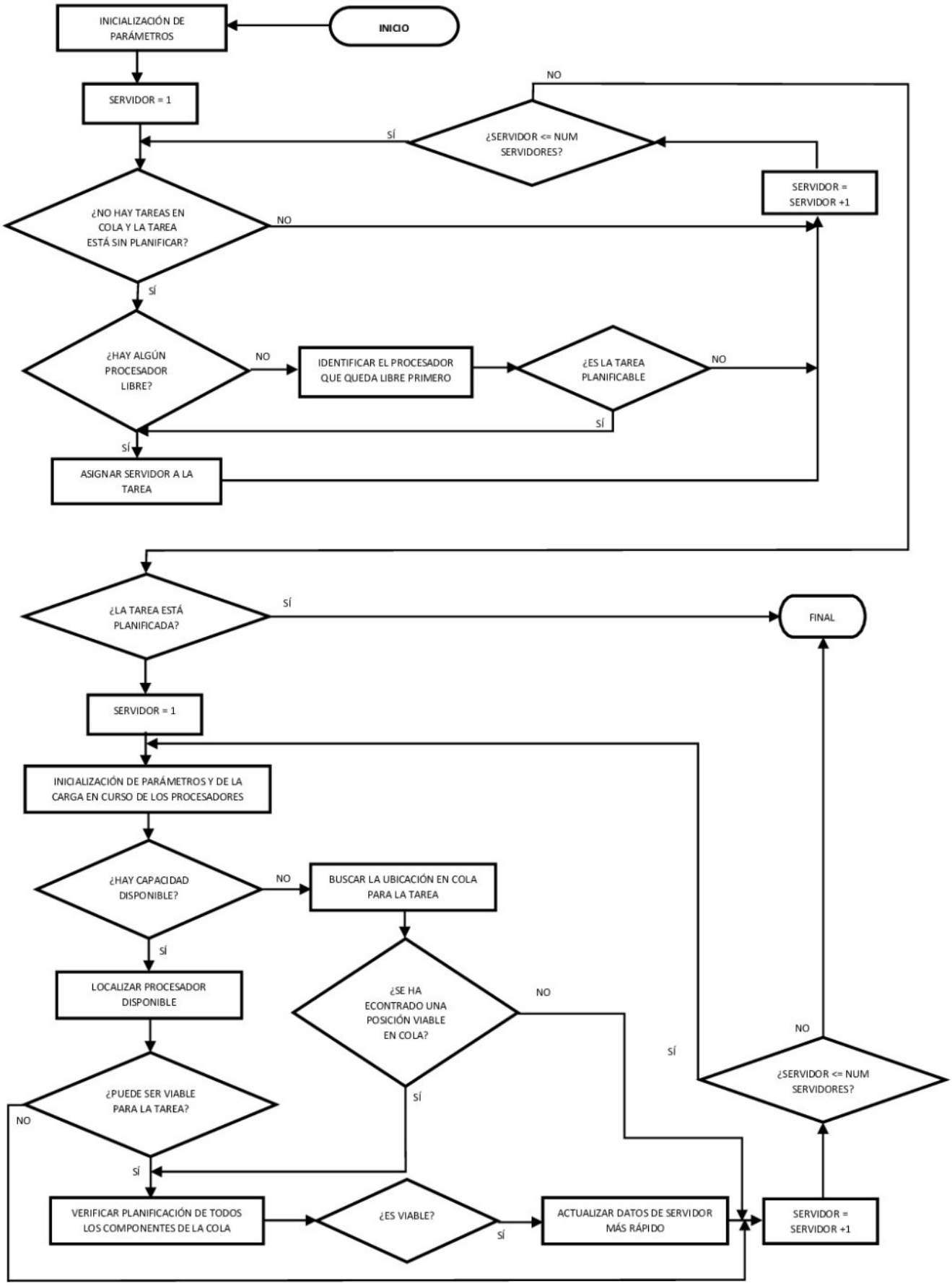

INICIO
INICIALIZACIÓN DE PARÁMETROS
SERVIDOR = 1
¿SERVIDOR <= NUM SERVIDORES?
SÍ
NO
SERVIDOR = SERVIDOR +1
¿NO HAY TAREAS EN COLA Y LA TAREA ESTÁ SIN PLANIFICAR?
NO
SÍ
¿HAY ALGÚN PROCESADOR LIBRE?
NO
IDENTIFICAR EL PROCESADOR QUE QUEDA LIBRE PRIMERO
¿ES LA TAREA PLANIFICABLE
NO
SÍ
SÍ
ASIGNAR SERVIDOR A LA TAREA
¿LA TAREA ESTÁ PLANIFICADA?
SÍ
FINAL
SERVIDOR = 1
INICIALIZACIÓN DE PARÁMETROS Y DE LA CARGA EN CURSO DE LOS PROCESADORES
¿HAY CAPACIDAD DISPONIBLE?
NO
BUSCAR LA UBICACIÓN EN COLA PARA LA TAREA
SÍ
LOCALIZAR PROCESADOR DISPONIBLE
¿SE HA ECONTRADO UNA POSICIÓN VIABLE EN COLA?
NO
SÍ
NO
¿SERVIDOR <= NUM SERVIDORES?
SÍ
¿PUEDE SER VIABLE PARA LA TAREA?
NO
SÍ
VERIFICAR PLANIFICACIÓN DE TODOS LOS COMPONENTES DE LA COLA
¿ES VIABLE?
SÍ
ACTUALIZAR DATOS DE SERVIDOR MÁS RÁPIDO
SERVIDOR = SERVIDOR +1

Figura 7.3

8. RESULTADOS Y COMPARATIVA

Una vez simuladas las planificaciones con los 3 diferentes algoritmos descritos en el punto anterior, y recopilados los datos extraídos de las mismas, se exponen los resultados a continuación.

Analizando en una primera instancia, los datos resultantes de la simulación de Arena, podremos comparar tanto longitud máxima de las colas de espera, como la espera media de las tareas en cola, así como el grado de ocupación medio de los servidores.

En la tabla 8.1 se presentan los datos referentes a longitudes máximas de cada cola de espera y los tiempos de espera medios en cada una de las colas. A la vista de estos resultados para EDF y LLF, se puede constatar que se cumple la premisa que establecimos durante la explicación del algoritmo de no utilizar cola de espera en los servidores. De igual manera, se confirma que en el algoritmo para planificación Myopic, tenemos restringida la longitud de cola de espera en los servidores a 7 tareas. De una manera muy general, podemos afirmar lo tiempos de espera medios parecen ser más competitivos en el caso de planificación EDF.

	EDF		**LLF**		**MYOPIC**	
	Long Max (ud)	Tespera medio (min)	Long Max (ud)	Tespera medio (min)	Long Max (ud)	Tespera medio (min)
Cola Principal	43	0.9186	49	1.271	28	0.4978
Cola Servidor 1	1	0	1	0	7	0.3269
Cola Servidor 2	1	0	1	0	7	2.2071

Tabla 8.1

Si verificamos el grado de utilización de los servidores (tabla 8.2), veremos un grado de utilización medio muy similar entre las 3 técnicas de planificación. Aparentemente, si evaluamos únicamente con ese dato, podríamos pensar que hay recursos libres durante la simulación, pero la aparición de colas de espera nos indica que en ciertos momentos, tenemos todos los recursos ocupados. Para entender más en profundidad la utilización de los recursos, se ha registrado la utilización de los recursos por hora y no por día. Este aspecto se comentará más adelante en esta memoria, con los gráficos 8.7, 8.8 y 8.9.

	EDF		LLF		MYOPIC	
	Uso Medio	Uso Procesadores Medio (ud)	Uso Medio	Uso Procesadores Medio (ud)	Uso Medio	Uso Procesadores Medio (ud)
Servidor 1	0.4789	3.3522	0.4858	3.4003	0.5288	3.7015
Servidor 2	0.4464	3.1247	0.4399	3.0795	0.3961	2.7724

Tabla 8.2

Además de los gráficos para las ocupaciones horarias de los recursos, nuestro código VBA registra datos de trabajos procesados, trabajos entregados a tiempo, tareas entregadas tarde, tareas expulsadas del sistema, adelanto medio en las entregas antes del plazo máximo y retraso medio en las entregas fuera de plazo. Estos datos se han registrado para cada servidor, y figuran en el Anexo C al final de esta memoria. De una forma más general, con la tabla 8.3, podemos intentar interpretar la información que podemos extraer de estos resultados, transformados en datos medios para el caso de los tiempos de adelanto o de retraso.

	EDF	LLF	MYOPIC
Tareas entregadas a tiempo	5610	5612	5612
Tareas entregadas tarde	8	8	8
Tareas expulsadas del sistema (total)	2	0	0
Earliness media (min)	8050,84	8047,62	8050,79
Lateness media (min)	21,72	21,87	23,04

Tabla 8.3

Según estos datos, los 3 métodos de planificación nos muestran resultados muy similares. Por la magnitud de los valores de adelanto (Earliness) y de retraso (Lateness), es más adecuado fijarnos en los valores de Lateness. En el caso de los valores de Earliness, si bien las diferencias son de varios minutos (en media), este valor es poco significativo respecto de los valores absolutos en términos de porcentaje.

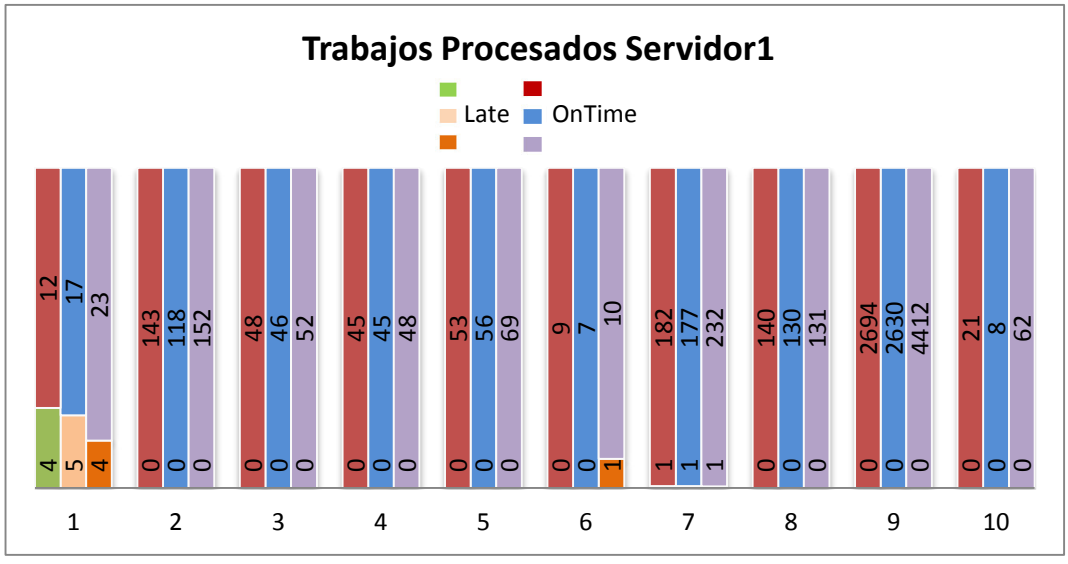
Además, siendo el objetivo conseguir una entrega de tareas dentro de un plazo máximo determinado, parece más interesante evaluar el impacto de los retrasos en las entregas. Concretando sobre el valor de Lateness (en media), vemos que la planificación EDF entrega los trabajos unos 10s menos tarde que la planificación LLF. Myopic quedaría en último lugar. No obstante, aunque la planificación EDF entregue menos tarde que la planificación LLF, vemos que hay dos tareas que han sido expulsadas por EDF, pero que han sido atendidas por LLF. Siendo que este es un sistema Soft Real-Time, parece adecuado pensar que una planificación LLF puede

dar buenos resultados en términos globales (EDF y LLF tienen retrasos de misma magnitud, pero LLF no necesita de la expulsión de tareas).

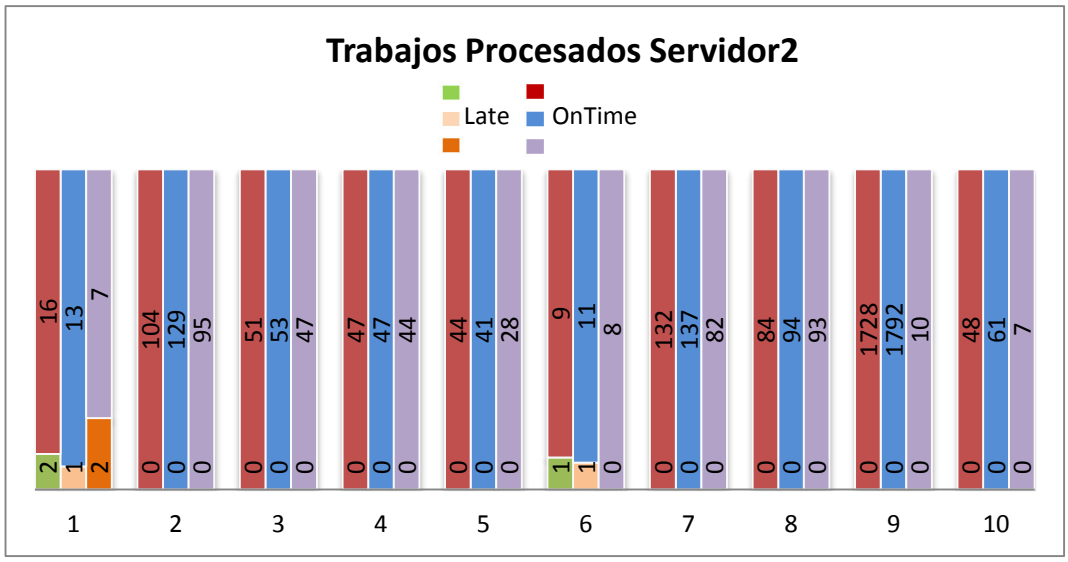
Veamos los resultados en términos de comparativa por cada concepto y para cada servidor:

Trabajos entregados dentro del plazo máximo establecido y trabajos entre fuera de plazo:

Podemos ver en las gráficas 8.1 y 8.2 adjuntas, dónde se localizan los trabajos que son entregados fuera del plazo máximo permitido. La concentración mayor se produce con los trabajos de tipo 1, seguidos por los trabajos de tipo 6, produciéndose las entregas fuera de plazo en ambos servidores. Una de las posibles causas podría ser el no disponer de suficientes datos para trabajar con unas estimaciones de duración más fiables. Ambos tipos de trabajo (1 y 6) son trabajos que tienen un plazo de entrega máximo de 20min, por lo que junto con la poca información disponible para las estimaciones, se complica el asegurar la entrega dentro del plazo. También vemos que hay entregas fuera de plazo en el caso de trabajos de tipo 7. Los trabajos de tipo 7 también tienen el plazo máximo más restrictivo, al igual que los de tipo 1 y tipo 6. En este caso, para la tarea 7, la discrepancia entre la estimación de duración y la duración real (que es no conocida) hace que la tarea se entregue tarde.



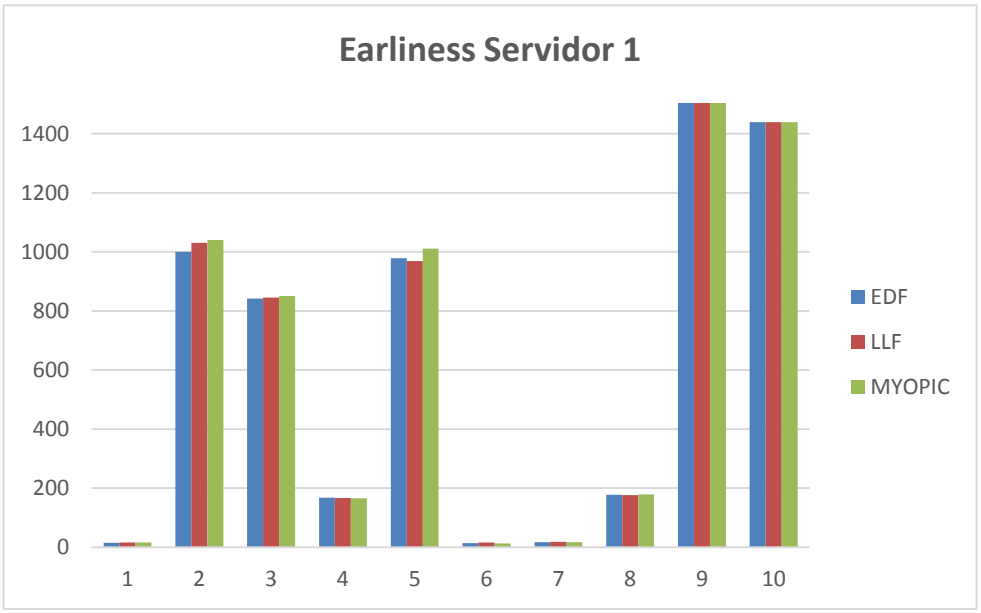
Gráfica 8.1



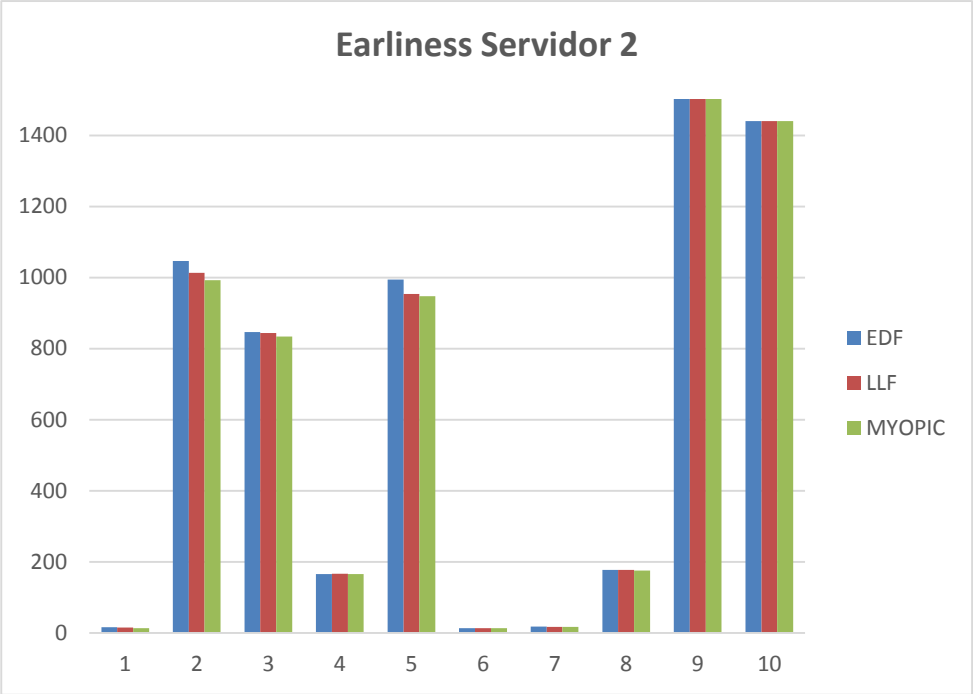
Gráfica 8.2

Media de adelanto en la entrega de los trabajos por tipo de trabajo y servidor:

En las gráfica 8.3 y 8.4, podemos ver que las medias de adelanto en minutos por tarea son muy similares entre los 3 tipos de planificación, destacando las diferencias que hay en los trabajos de tipo 2 y tipo 5. En función del servidor, el adelanto en la entrega es mayor con planificación EDF, o con planificación Myopic. Hemos visto antes, que pese a la diferencia que puede apreciarse gráficamente, estas diferencias son despreciables si comparamos las diferencias de adelantos entre los distintos tipos de planificación con los valores de adelanto absolutos.



Gráfica 8.3



Gráfica 8.4

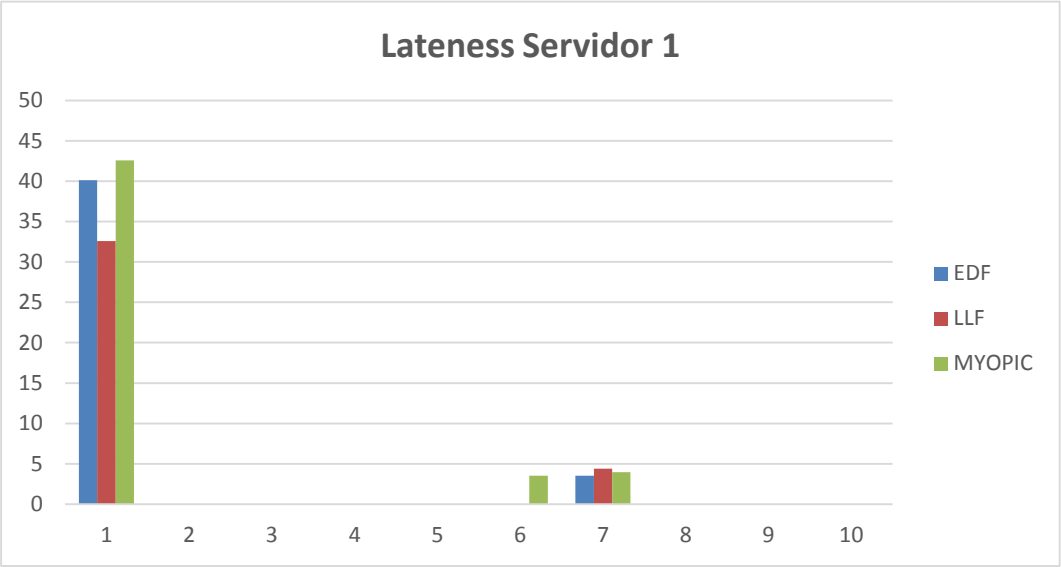
No hay posibilidad de decidir claramente, en función de este concepto, si alguno de los métodos de planificación probados es mejor que los demás. Quizás, por simplicidad de programación y porque debemos recordar todos los condicionantes de no conocimiento de previsión de llegadas, ni tipos de trabajo, ni duración de los mismos, tenderemos a seleccionar una planificación EDF.

Media de retraso en la entrega de los trabajos por tipo de trabajo y servidor:

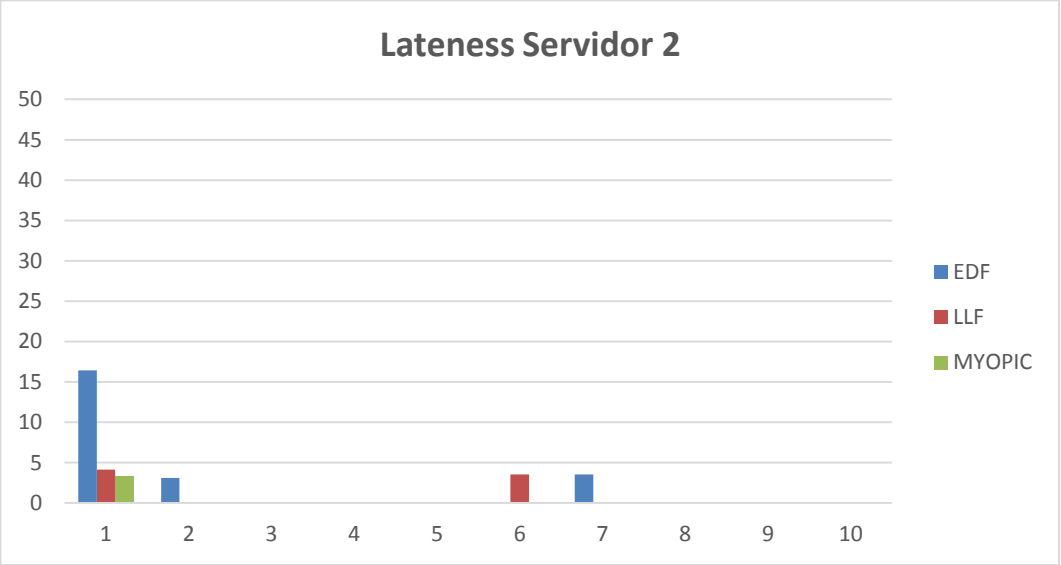
En las gráficas 8.5 y 8.6, vemos las medias de retraso en minutos para cada tipo de tarea y cada servidor. Estos datos únicamente afectan a los trabajos que se han procesado y entregado fuera de plazo, como hemos visto en las gráficas 8.1 y 8.2.

Dado que nuestro código programado verifica si la tarea puede ser entregada en plazo antes de enviarla al servidor asignado, podemos estar seguros que el retraso es originado por la discrepancia entre la estimación de duración realizada y la duración real que ha tenido el trabajo una vez entrado en el servidor.

Los valores de las gráficas nos indican, como ya vimos anteriormente, que, en este aspecto, la planificación LLF sea más adecuada si valoramos que las tareas que se entregan tarde, se entreguen con un retraso lo más mínimo u óptimo posible.



Gráfica 8.5



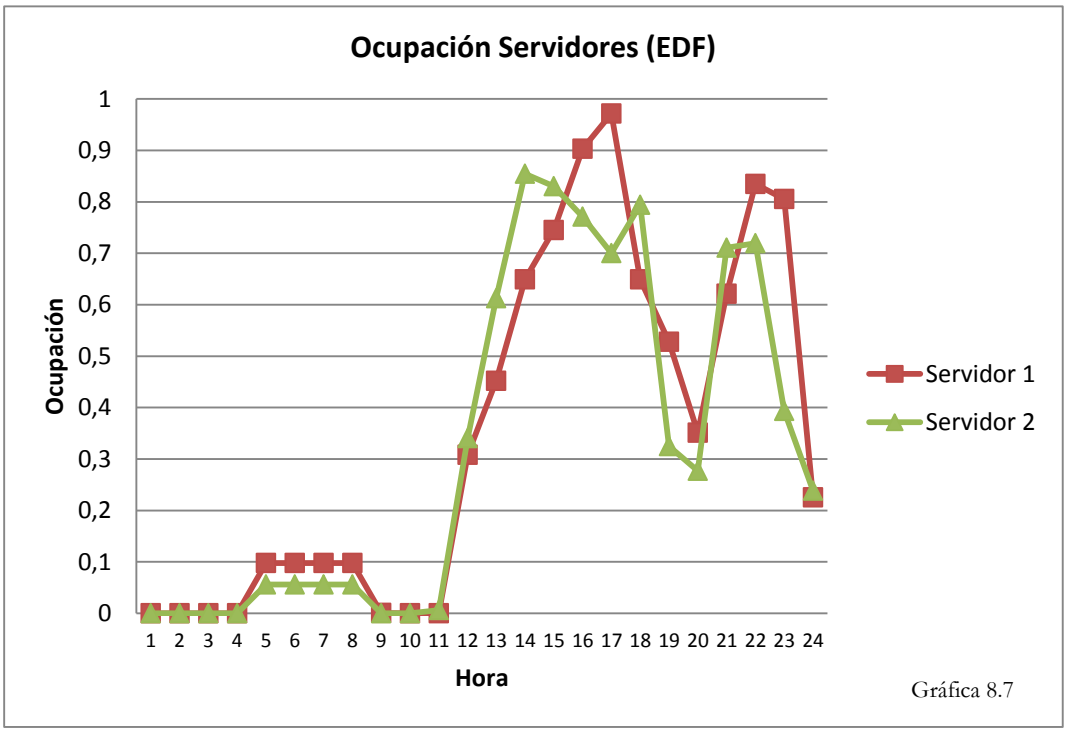
Gráfica 8.6

Ocupación por hora de cada servidor:

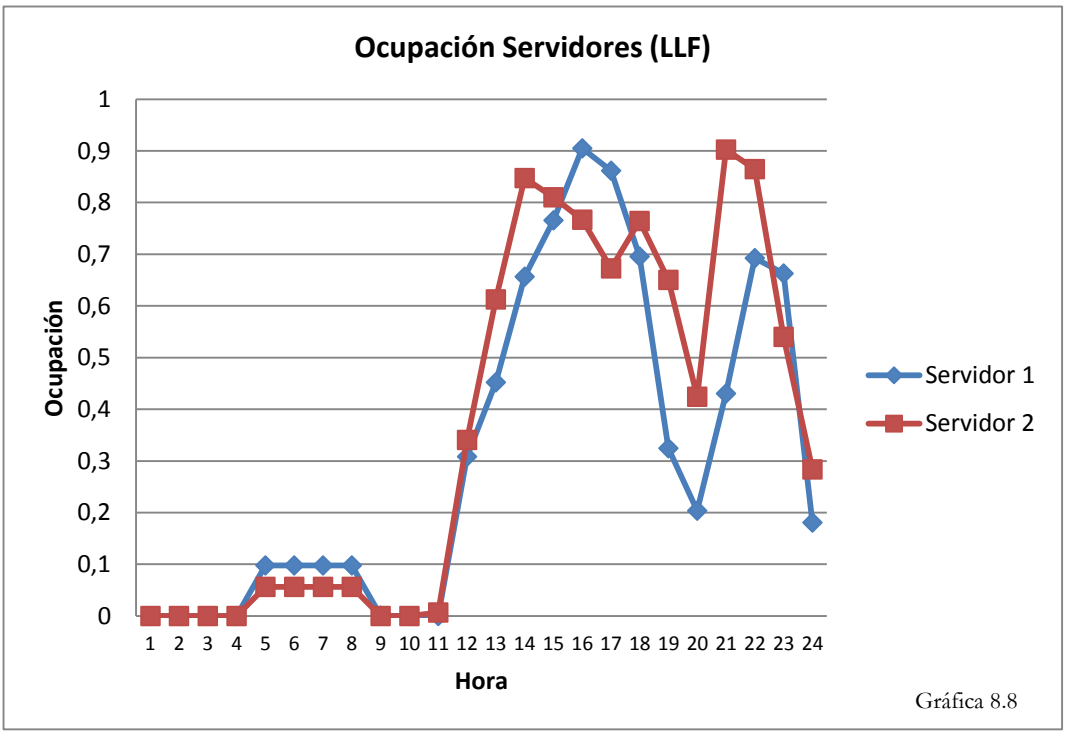
Durante las simulaciones, se ha registrado el uso medio horario de cada servidor. El resultado puede verse en las gráficas 8.7, 8.8 y 8.9 a continuación, dónde se indica la utilización en tanto por uno frente al eje horario.

A la vista de las gráficas para cada uno de los tipos de planificación, puede constatarse que tienen una utilización horaria análoga en todos los casos. El aspecto interesante que podemos destacar es que se aprecian de forma clara unas zonas

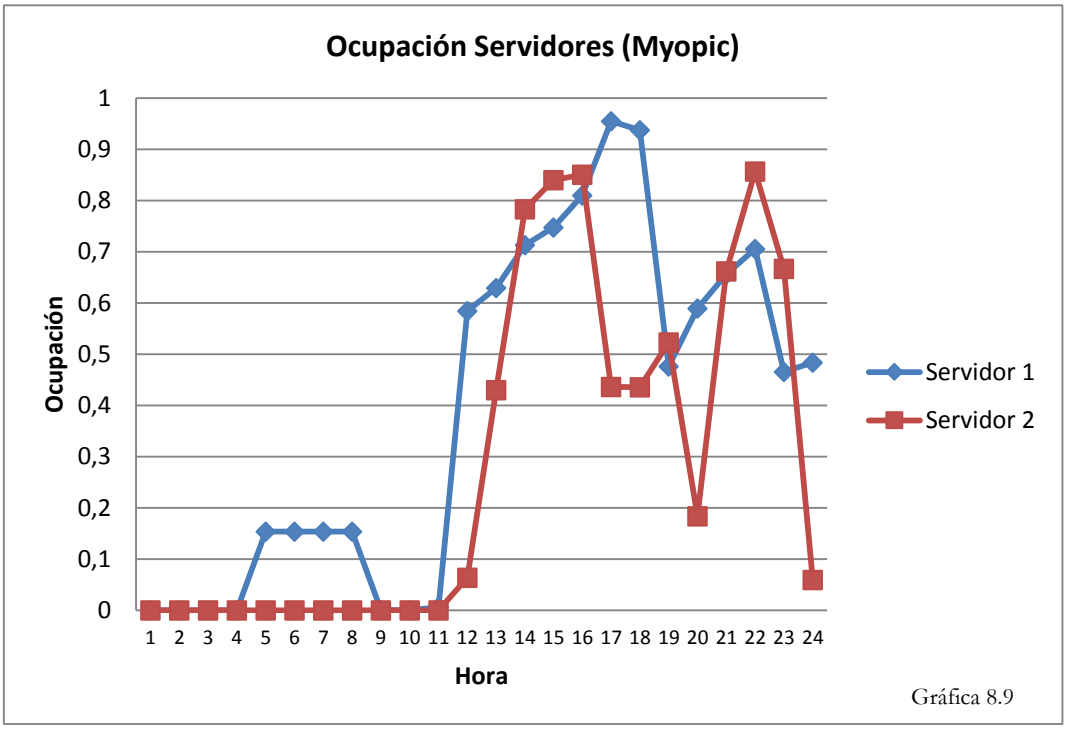
horarias dónde los servidores están infrautilizados (de 0h a 11h). Esto nos induce a pensar que quizás se puedan utilizar otros tipos de planificación que nos permitan distribuir de otra manera los trabajos, pudiendo realizar por la noche aquellos que tuvieran un plazo máximo mayor. De esta manera, podríamos liberar capacidad de cálculo en otros horarios más comprometidos.



Gráfica 8.7



Gráfica 8.8


Ocupación Servidores (Myopic)
Ocupación
1
0,9
0,8
0,7
0,6
0,5
0,4
0,3
0,2
0,1
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Hora
Servidor 1
Servidor 2

Gráfica 8.9

9. CONCLUSIONES Y RECOMENDACIONES

Una vez estudiados los resultados obtenidos de las simulaciones para cada uno de los tipos de planificación escogidos, y teniendo en cuenta las condiciones iniciales de planteamiento de problema, podemos llegar a la conclusión que un método sencillo de planificación en este caso podría ser suficiente para obtener unos resultados más que satisfactorios.

Este método sencillo de planificación podría ser bien EDF o bien LLF. Myopic es un método más complejo en lo referente a programación, y para nuestro caso, no ha aportado beneficios respecto a las otras dos opciones.

Dadas las condiciones de definición de los trabajos (no conocemos cuando llegan al sistema, ni su duración), será más útil o robusto utilizar el criterio EDF que el LLF, dado que este último utiliza las estimaciones de duración para fijar las prioridades de asignación de servidor. Este aspecto aporta más incertidumbre a LLF que a EDF, que tan sólo utiliza el plazo máximo permitido para establecer las prioridades. De esta forma, limitaremos el uso de las estimaciones de duración calculada a la verificación que se realiza antes de enviar las tareas a la cola local o al servidor para comprobar que puede ser finalizada dentro de plazo máximo autorizado.

En base a lo visto hasta ahora, podrían abrirse al menos 3 vías de estudio como posibilidades de mejora de rendimiento de estos algoritmos de planificación.

La primera vía, a la vista de las gráficas de utilización horaria, sería desplazar la ejecución de algunos tipos de trabajo a zonas horarias con baja ocupación de servidor, intentando cubrir la zona de 0h a 11h con tareas del tipo 2, 3, 5, 9 y/o 10 por ejemplo. Estas tareas son las que nos podrían dar mayor flexibilidad desde el punto de vista de plazo máximo permitido (ver tabla 6.1).

La segunda vía podría ser intentar mejorar la estimación de cálculo de duraciones, pasando el mínimo de datos disponibles para empezar a utilizar estimaciones de 15 trabajos del mismo tipo a 30 trabajos o superior. Esto mejorará la calidad de las estimaciones de duración, sin ser, claro está, una solución definitiva.

La tercera vía podría priorizar las tareas con plazos de entrega máximo de 20min sobre las demás. Habría que hacer la simulación detallada, para ver cómo esta prioridad añadida afecta al cumplimiento de los plazos del resto de trabajos.

Otra posibilidad que se podría haber valorado, pero que no se recomienda, es la dedicar algún servidor, o algunos procesadores, en exclusiva, a servir las tareas con

plazo de entrega corto (20min). No es recomendable en el entorno de nuestro caso, puesto que no hay forma de definir cuántos procesadores o cuántos servidores serían los adecuados. Recordemos una vez más, que no conocemos los tiempos de llegada de las tareas, ni sus duraciones, por lo que no podemos prever con la exactitud la necesidad de recursos.

Una recomendación adicional puede ser utilizar estos algoritmos, tal cual están ahora mismo o mejorados, para intentar optimizar también la cantidad de recursos necesarios. Al final, con una buena planificación, quizás podría ahorrarse el uso de algún procesador que podría quedar en reserva para situaciones excepcionales.

10. PRESUPUESTO DEL TFM

Este trabajo de final de máster tiene un presupuesto para su realización que constará de los conceptos siguientes:

CONCEPTO	Ctd (h)	Coste (Eur/h)	(Eur)
M.O Ingeniería	325	50	16250
Amortización Equipo Informático	240	20	4800
Total			21050

Tabla 10.1

11. PLANIFICACIÓN Y PROGRAMACIÓN DEL TFM

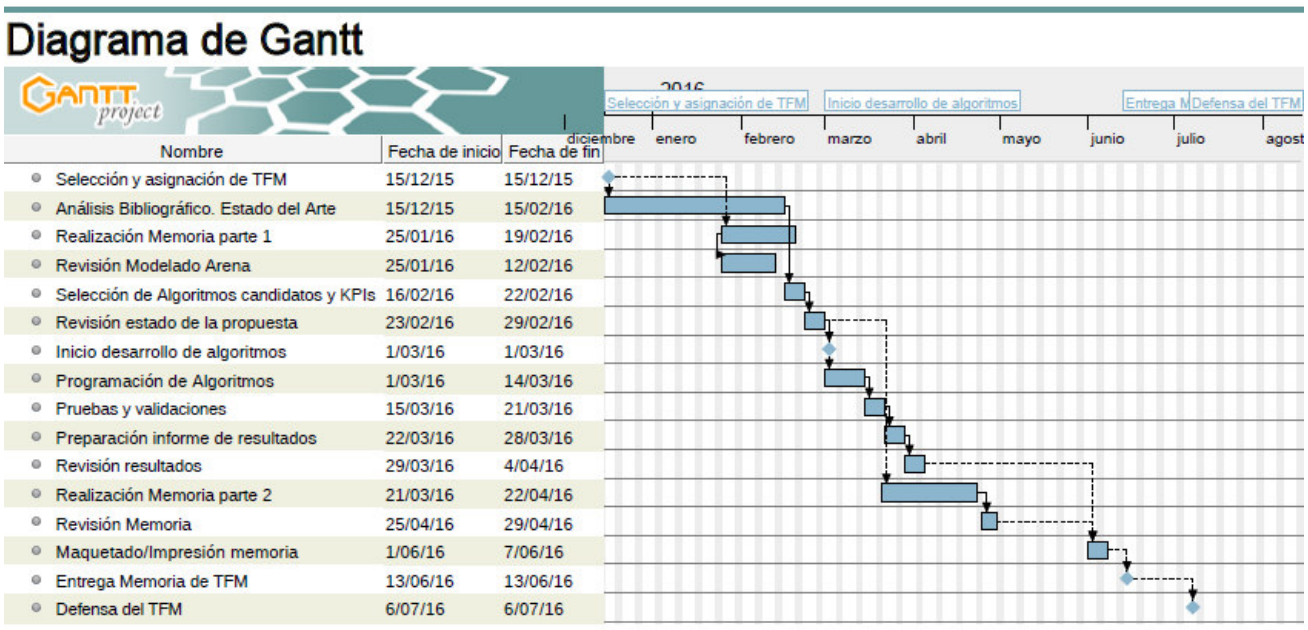
La planificación y programación del Trabajo de Final de Máster (TFM) se ha llevado en base a 5 fases diferenciadas:

1. Asignación de TFM
2. Investigación estado del arte
3. Selección de algoritmos candidatos e indicadores
4. Programación y resultados
5. Revisión resultados y redacción de memoria

De forma transversal, durante todo el tiempo de desarrollo, se han mantenido reuniones periódicas con el director del TFM para revisión del estado del trabajo, de

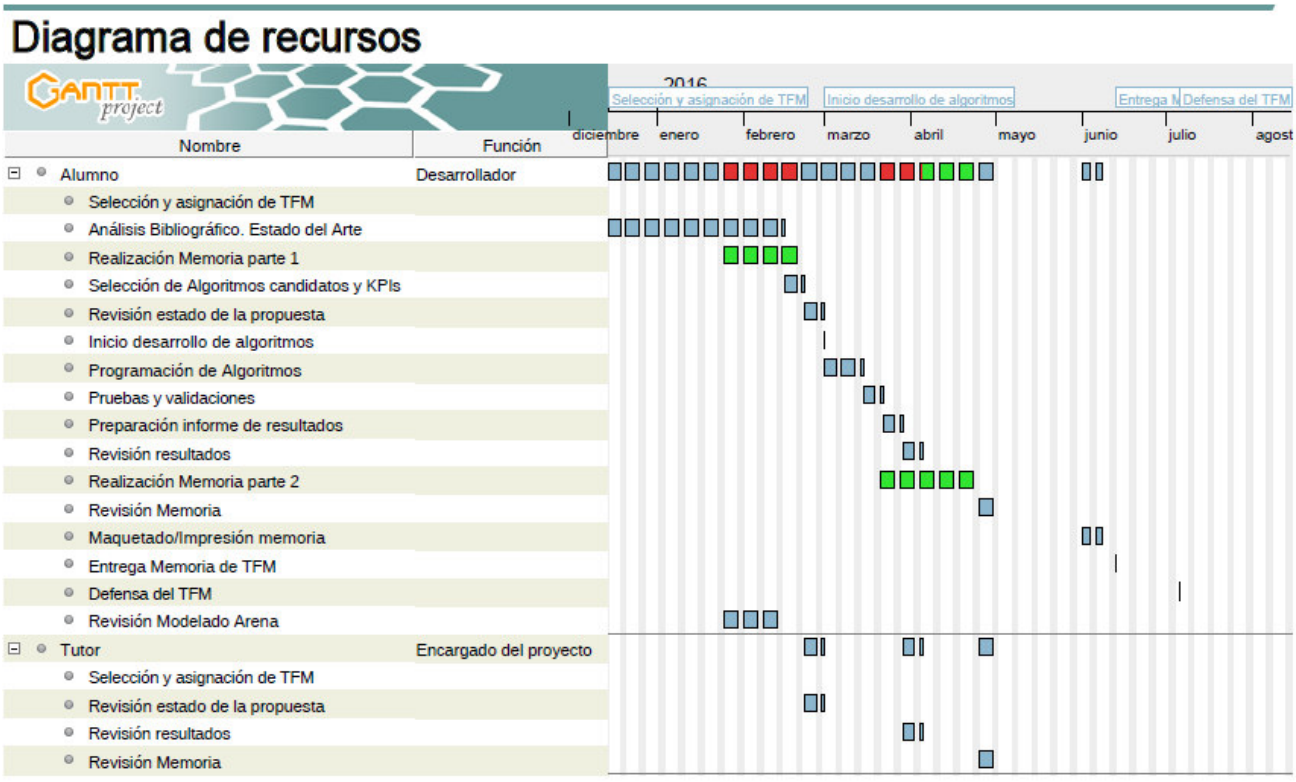
los objetivos concretos si necesario, y confirmación de la correcta evolución de los resultados parciales.

Todos estos eslabones pueden verse en la planificación de proyecto a continuación. También se ha incluido una planificación de recursos necesaria para más información.



Nombre	Fecha de inicio	Fecha de fin
Selección y asignación de TFM	15/12/15	15/12/15
Análisis Bibliográfico. Estado del Arte	15/12/15	15/02/16
Realización Memoria parte 1	25/01/16	19/02/16
Revisión Modelado Arena	25/01/16	12/02/16
Selección de Algoritmos candidatos y KPIs	16/02/16	22/02/16
Revisión estado de la propuesta	23/02/16	29/02/16
Inicio desarrollo de algoritmos	1/03/16	1/03/16
Programación de Algoritmos	1/03/16	14/03/16
Pruebas y validaciones	15/03/16	21/03/16
Preparación informe de resultados	22/03/16	28/03/16
Revisión resultados	29/03/16	4/04/16
Realización Memoria parte 2	21/03/16	22/04/16
Revisión Memoria	25/04/16	29/04/16
Maquetado/Impresión memoria	1/06/16	7/06/16
Entrega Memoria de TFM	13/06/16	13/06/16
Defensa del TFM	6/07/16	6/07/16

Gráfico 11.1: Planificación de proyecto



Nombre	Función
Alumno	Desarrollador
Selección y asignación de TFM	
Análisis Bibliográfico. Estado del Arte	
Realización Memoria parte 1	
Selección de Algoritmos candidatos y KPIs	
Revisión estado de la propuesta	
Inicio desarrollo de algoritmos	
Programación de Algoritmos	
Pruebas y validaciones	
Preparación informe de resultados	
Revisión resultados	
Realización Memoria parte 2	
Revisión Memoria	
Maquetado/Impresión memoria	
Entrega Memoria de TFM	
Defensa del TFM	
Revisión Modelado Arena	
Tutor	Encargado del proyecto
Selección y asignación de TFM	
Revisión estado de la propuesta	
Revisión resultados	
Revisión Memoria	

Gráfico 11.2: Planificación de recursos

12. BIBLIOGRAFÍA

[1] Lindh, F., Otnes, T., & Wennerström, J. **Scheduling Algorithms for Real-Time Systems.** *Department of Computer Engineering, Malardalens University, Sweden*.

[2] Annette J, R., Banu W, A., & Shriram, S. (2013**). A Taxonomy and Survey of Scheduling Algorithms in Cloud: Based on task dependency**. *International Journal of Computer Applications*, *82*(15), 20-26.

[3] Iqbal, A., Zafar, A., & Siddique, B. (2005, September). **Dynamic Queue Deadline First scheduling algorithm for soft real-time System**. In *IEEE International Conference on Emerging Technologies, Islamabad* (pp. 346-351).

[4] Ramamritham, K., Stankovic, J. A., & Shiah, P. F. (1990). **Efficient scheduling algorithms for real-time multiprocessor systems**. *Parallel and Distributed Systems, IEEE Transactions on*, *1*(2), 184-194.

[5] Gregg, C., Boyer, M., Hazelwood, K., & Skadron, K. (2011, June). **Dynamic heterogeneous scheduling decisions using historical runtime data**. In *Workshop on Applications for Multi-and Many-Core Processors (A4MMC)*.

[6] Megow, N., Uetz, M., & Vredeveld, T. (2006). **Models and algorithms for stochastic online scheduling**. *Mathematics of Operations Research*, *31*(3), 513-525.

[7] Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., Reuther, A. I., ... & Freund, R. F. (2001**). A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems**. *Journal of Parallel and Distributed computing*, *61*(6), 810-837.

[8] Armstrong, R., Hensgen, D., & Kidd, T. (1998, March). **The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions**. In *Heterogeneous Computing Workshop, 1998.(HCW 98) Proceedings. 1998 Seventh* (pp. 79-87). IEEE.

[9] Maheswaran, M., Ali, S., Siegal, H. J., Hensgen, D., & Freund, R. F. (1999). **Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems**. In *Heterogeneous Computing Workshop, 1999.(HCW'99) Proceedings. Eighth* (pp. 30-44). IEEE.

[10] Casanova, H., Legrand, A., Zagorodnov, D., & Berman, F. (2000). **Heuristics for scheduling parameter sweep applications in grid environments**.

In *Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th* (pp. 349-363). IEEE.

[11] Bertsimas, D., & Tsitsiklis, J. (1993). **Simulated annealing**. *Statistical science*, *8*(1), 10-15.

[12] Burns, A., & Audsley N. (1995), **Real-Time System Scheduling**. *Predicatably Dependable Computer Systems*, *2*(2) - II, 10-31. Or Department of Computer Science, University of York, UK.

[13] Dertouzos, M. L., & Mok, A. K. L. (1989). **Multiprocessor online scheduling of hard-real-time tasks**. *IEEE Transactions on Software Engineering, 15*(12), 1497-1506.

ANEXOS

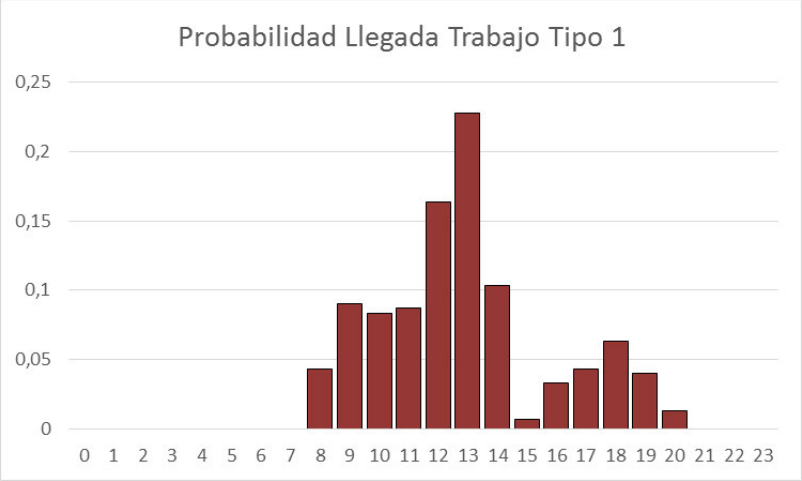
ANEXO A: Distribuciones de probabilidad de llegada para cada tipo de trabajo

En este anexo se presentan, para cada tipo de solicitud trabajo, su distribución de probabilidad de llegada al sistema, en función de la hora en la que se encuentre la simulación.

Basándonos en las tomas de datos existentes, se realiza un conteo del número de llegadas que ha habido en cada hora transcurrida. Con este conteo, es sencillo encontrar la probabilidad que le corresponde a esa hora, dividiendo el número de sucesos en esa hora por el número de sucesos totales.

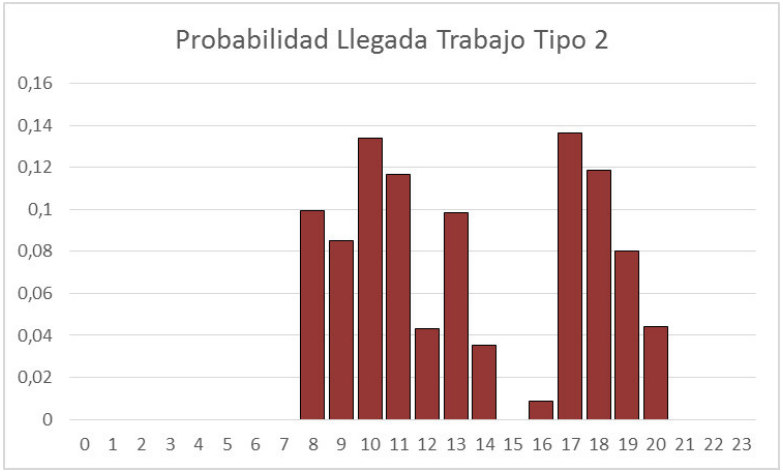
Exponemos a continuación cada tipo de trabajo con la tabla correspondiente a los cálculos y una representación gráfica de la probabilidad de llegada mediante histograma.

Tipo de trabajo		1
Intervalo (hora)	Nº de Sucesos en Intervalo	Probabilidad
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	13	0,043478
9	27	0,090301
10	25	0,083612
11	26	0,086957
12	49	0,163880
13	68	0,227425
14	31	0,103679
15	2	0,006689
16	10	0,033445
17	13	0,043478
18	19	0,063545
19	12	0,040134
20	4	0,013378
21	0	0
22	0	0
23	0	0
Total	299	



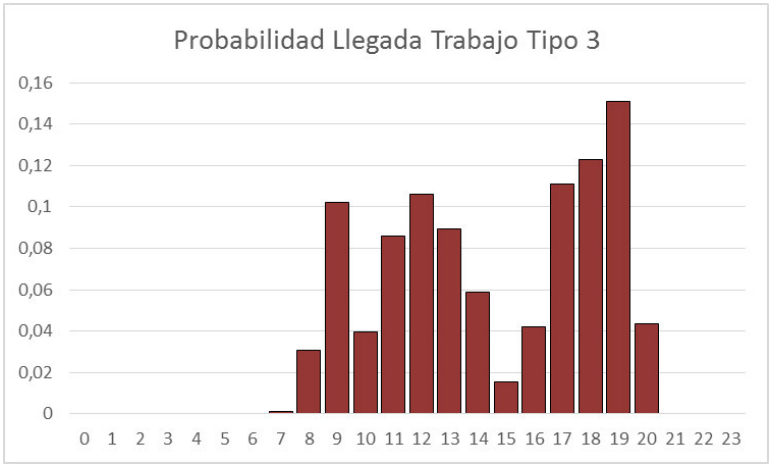
Tabla/Gráfico A.1

Tipo de trabajo		2
Intervalo (hora)	Nº de Sucesos en Intervalo	Probabilidad
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	169	0,099646
9	144	0,084906
10	227	0,133844
11	198	0,116745
12	73	0,043042
13	167	0,098467
14	60	0,035377
15	0	0
16	15	0,008844
17	231	0,136203
18	201	0,118514
19	136	0,080189
20	75	0,044222
21	0	0
22	0	0
23	0	0
Total	1696	



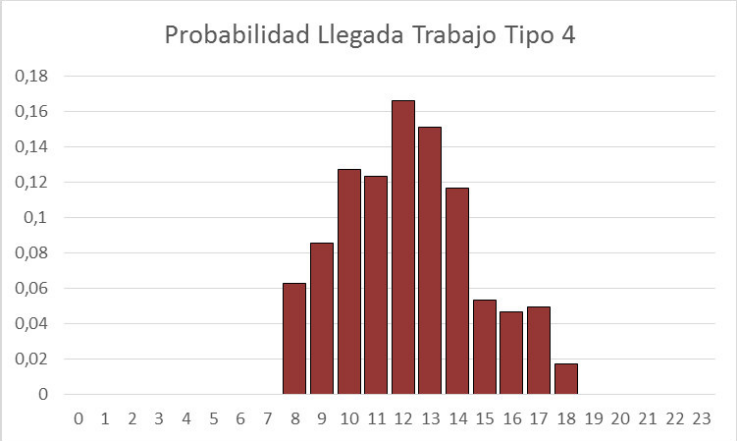
Tabla/Gráfico A.2

Tipo de trabajo		3
Intervalo (hora)	Nº de Sucesos en Intervalo	Probabilidad
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	1	0,001279
8	24	0,030691
9	80	0,102302
10	31	0,039642
11	67	0,085678
12	83	0,106138
13	70	0,089514
14	46	0,058824
15	12	0,015345
16	33	0,042199
17	87	0,111253
18	96	0,122762
19	118	0,150895
20	34	0,043478
21	0	0
22	0	0
23	0	0
Total	782	



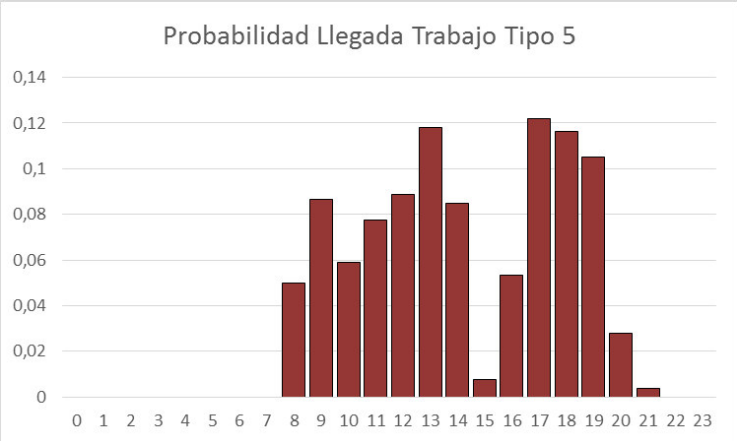
Tabla/Gráfico A.3

Tipo de trabajo		4
Intervalo (hora)	Nº de Sucesos en Intervalo	Probabilidad
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	47	0,062918
9	64	0,085676
10	95	0,127175
11	92	0,123159
12	124	0,165997
13	113	0,151272
14	87	0,116466
15	40	0,053548
16	35	0,046854
17	37	0,049531
18	13	0,017403
19	0	0
20	0	0
21	0	0
22	0	0
23	0	0
Total	747	



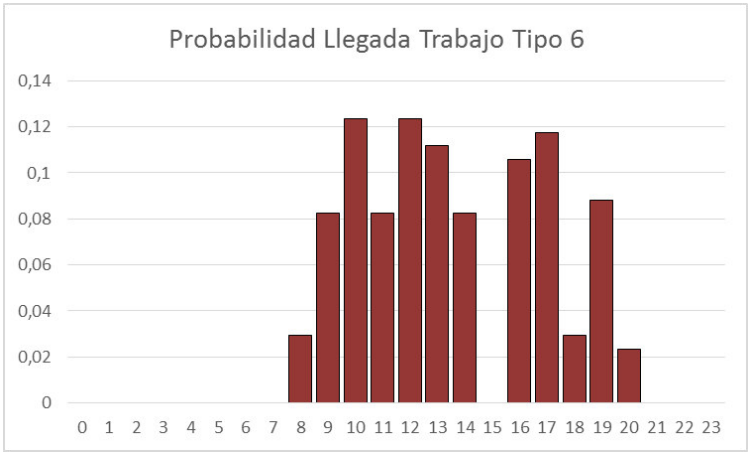
Tabla/Gráfico A.4

Tipo de trabajo		5
Intervalo (hora)	Nº de Sucesos en Intervalo	Probabilidad
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	27	0,049815
9	47	0,086716
10	32	0,059041
11	42	0,077491
12	48	0,088561
13	64	0,118081
14	46	0,084871
15	4	0,007380
16	29	0,053506
17	66	0,121771
18	63	0,116236
19	57	0,105166
20	15	0,027675
21	2	0,003690
22	0	0
23	0	0
Total	542	



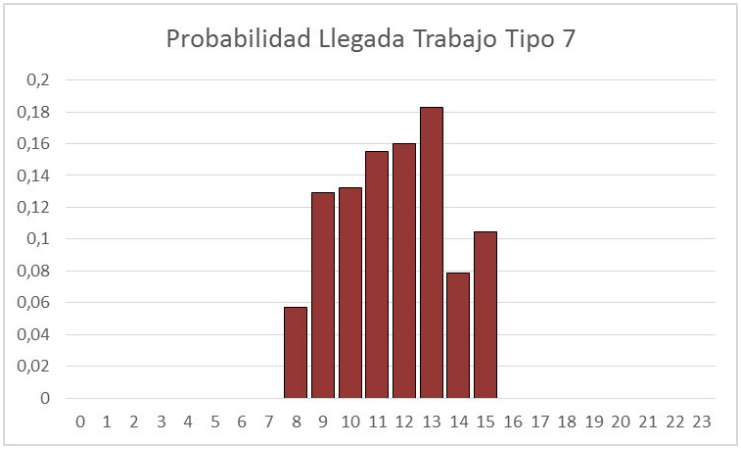
Tabla/Gráfico A.5

Tipo de trabajo		6
Intervalo (hora)	Nº de Sucesos en Intervalo	Probabilidad
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	5	0,029412
9	14	0,082353
10	21	0,123529
11	14	0,082353
12	21	0,123529
13	19	0,111765
14	14	0,082353
15	0	0
16	18	0,105882
17	20	0,117647
18	5	0,029412
19	15	0,088235
20	4	0,023529
21	0	0
22	0	0
23	0	0
Total	170	



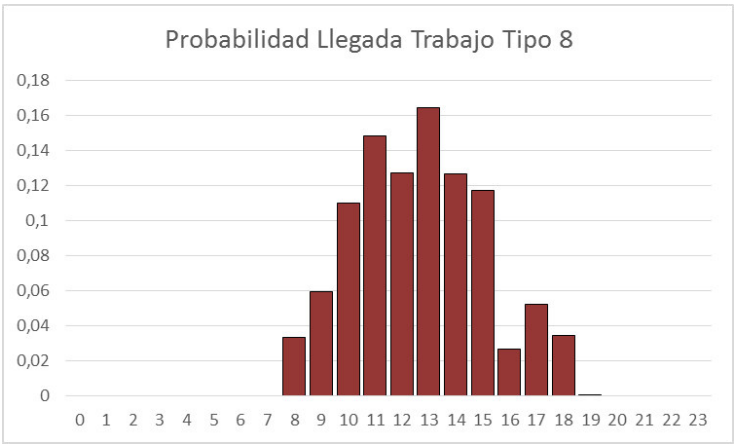
Tabla/Gráfico A.6

Tipo de trabajo		7
Intervalo (hora)	Nº de Sucesos en Intervalo	Probabilidad
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	130	0,057168
9	294	0,129288
10	301	0,132366
11	353	0,155233
12	364	0,160070
13	416	0,182938
14	179	0,078716
15	237	0,104222
16	0	0
17	0	0
18	0	0
19	0	0
20	0	0
21	0	0
22	0	0
23	0	0
Total	2274	



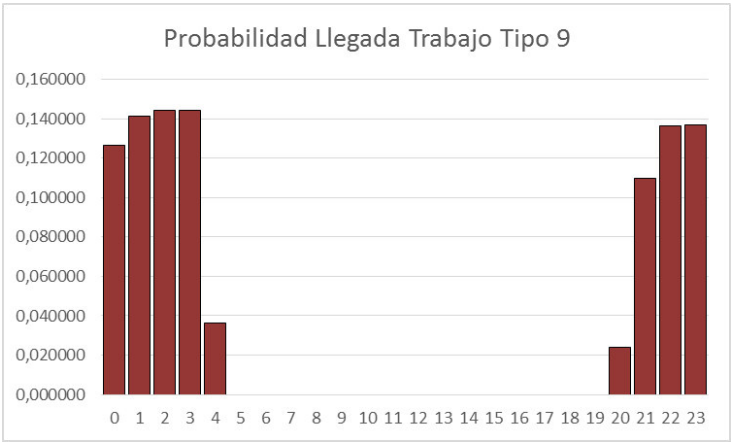
Tabla/Gráfico A.7

Tipo de trabajo		8
Intervalo (hora)	Nº de Sucesos en Intervalo	Probabilidad
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	43	0,033541
9	76	0,059282
10	141	0,109984
11	190	0,148206
12	163	0,127145
13	211	0,164587
14	162	0,126365
15	150	0,117005
16	34	0,026521
17	67	0,052262
18	44	0,034321
19	1	0,000780
20	0	0
21	0	0
22	0	0
23	0	0
Total	1282	



Tabla/Gráfico A.8

Tipo de trabajo		9
Intervalo (hora)	Nº de Sucesos en Intervalo	Probabilidad
0	3852	0,126482
1	4299	0,141159
2	4399	0,144443
3	4393	0,144246
4	1108	0,036382
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
19	0	0
20	735	0,024134
21	3340	0,109670
22	4158	0,136529
23	4171	0,136956
Total	30455	



Tabla/Gráfico A.9

Tipo de trabajo		10
Intervalo (hora)	Nº de Sucesos en Intervalo	Probabilidad
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	0
10	0	0
11	0	0
12	0	0
13	0	0
14	0	0
15	0	0
16	0	0
17	0	0
18	0	0
19	0	0
20	179	0,335206
21	355	0,664794
22	0	0
23	0	0
Total	534	

Tabla/Gráfico A.10

ANEXO B: Código fuente de los algoritmos

En este anexo se presenta el código fuente desarrollado para cada uno de los métodos utilizados durante el presente TFM.

En cada caso se expondrá el esquema de funcionamiento, junto con los procedimientos programados para cada caso.

En cada activación de simulación, se inicializarán unos parámetros globales que definen el entorno mediante el procedimiento expuesto a continuación. Esto son los parámetros que aparecen más abajo:

varAlgoritmo: Asignando el valor 1, 2 o 3 estaremos fijando el método de planificación que queremos utilizar durante la ejecución; siendo EDF, LLF y Myopic respectivamente.

varNumServidores: Fija el número de servidores a 2.

varNumCores: Fija el número de procesadores a 7. Es un parámetro utilizado fundamentalmente en una subrutina del método Myopic para la búsqueda de la mejor propuesta de planificación para un trabajo

varNumTrabajos: Fija el número de tipos de trabajos. En nuestro caso disponemos de 10 tipos de trabajo en todo momento.

```
Private Sub ModelLogic_RunBeginSimulation()

  Dim m As Model
  Dim s As Arena.SIMAN
  Dim xl As Object
  Dim filetoopen As String
  Dim arenadir As String
  Dim varCounter As String 'rango de las variables
  Dim variables As Integer 'numero de las variables
  Dim Rtmp As Variant
  Dim nomvariable As String
  Dim i As Integer, flg As Boolean

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  s.VariableArrayValue(s.SymbolNumber("varAlgoritmo")) = 1
  s.VariableArrayValue(s.SymbolNumber("varNumServidores")) = 2
  s.VariableArrayValue(s.SymbolNumber("varNumCores")) = 7
  s.VariableArrayValue(s.SymbolNumber("varNumTrabajos")) = 10
```

```
    Open "TESTFILE" For Output As #1 ' Open file for output.

End Sub
```

B.1: Algoritmo EDF

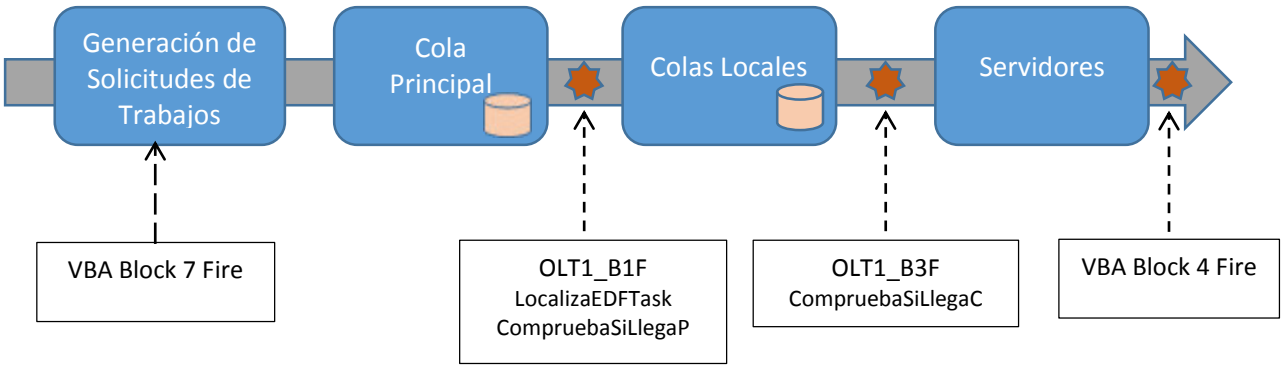


Gráfico B.1

Los procedimientos principales desencadenados por los eventos durante la simulación con planificación **EDF** son los siguientes:

```
Private Sub VBA_Block_7_Fire()

  '* Evento periodico para la generación de las solicitudes de trabajo

  Dim m As Model
  Dim s As Arena.SIMAN
  Dim val1 As Double, prob As Double, varHora As Integer
  Dim newEntity As Long, estacionColaPrincipal As Long
  Dim i As Integer, resto As Double

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  varHora = s.VariableArrayValue(s.SymbolNumber("varHora"))
  '... El nom de la estacio es el nom que surt en el camp "Station name"
  estacionColaPrincipal = s.SymbolNumber("Station Cola Principal")

  For i = 1 To 10
    prob = s.VariableArrayValue(s.SymbolNumber("varProbLlegadas", varHora + 1, i))
    val1 = s.SampleUniform(0#, 1#, 1)
    If val1 <= prob Then
      newEntity = s.EntityCreate
      '... Asignamos el tipo de trabajo
      s.EntityAttribute(newEntity, s.SymbolNumber("atTipoTrabajo")) = i
      '... Asignamos el tiempo de proceso
      If i = 1 Then
        s.EntityAttribute(newEntity, s.SymbolNumber("atTtrabajo")) = (s.SampleLognormal(400, 1350, 2)) / 60#
        s.EntityAttribute(newEntity, s.SymbolNumber("atTlimite")) = s.RunCurrentTime + 20
```

```
    ElseIf i = 2 Then
      s.EntityAttribute(newEntity, s.SymbolNumber("atTtrabajo")) = (s.SampleLognormal(51, 57.4, 2) * 6) / 60#
      resto = s.RunCurrentTime Mod 1440
      If resto < 420 Then
        s.EntityAttribute(newEntity, s.SymbolNumber("atTlimite")) = s.RunCurrentTime + 420 - resto
      Else
        s.EntityAttribute(newEntity, s.SymbolNumber("atTlimite")) = s.RunCurrentTime + (1440 - resto) + 420
      End If
    ElseIf i = 3 Then
      s.EntityAttribute(newEntity, s.SymbolNumber("atTtrabajo")) = s.SampleTriangular(5#, 460#, 9100#, 2) / 60#
      resto = s.RunCurrentTime Mod 1440
      If resto < 360 Then
        s.EntityAttribute(newEntity, s.SymbolNumber("atTlimite")) = s.RunCurrentTime + 360 - resto
      Else
        s.EntityAttribute(newEntity, s.SymbolNumber("atTlimite")) = s.RunCurrentTime + (1440 - resto) + 360
      End If
    ElseIf i = 4 Then
      s.EntityAttribute(newEntity, s.SymbolNumber("atTtrabajo")) = s.SampleTriangular(4#, 106#, 2040, 2) / 60#
      s.EntityAttribute(newEntity, s.SymbolNumber("atTlimite")) = s.RunCurrentTime + 180
    ElseIf i = 5 Then
      s.EntityAttribute(newEntity, s.SymbolNumber("atTtrabajo")) = s.SampleLognormal(34, 29.5, 2) / 60#
      resto = s.RunCurrentTime Mod 1440
      If resto < 420 Then
        s.EntityAttribute(newEntity, s.SymbolNumber("atTlimite")) = s.RunCurrentTime + 420 - resto
      Else
        s.EntityAttribute(newEntity, s.SymbolNumber("atTlimite")) = s.RunCurrentTime + (1440 - resto) + 420
      End If
    ElseIf i = 6 Then
      s.EntityAttribute(newEntity, s.SymbolNumber("atTtrabajo")) = s.SampleExponential(331, 2) / 60#
      s.EntityAttribute(newEntity, s.SymbolNumber("atTlimite")) = s.RunCurrentTime + 20
    ElseIf i = 7 Then
      s.EntityAttribute(newEntity, s.SymbolNumber("atTtrabajo")) = s.SampleLognormal(96, 127, 2) / 60#
      s.EntityAttribute(newEntity, s.SymbolNumber("atTlimite")) = s.RunCurrentTime + 20
    ElseIf i = 8 Then
      s.EntityAttribute(newEntity, s.SymbolNumber("atTtrabajo")) = s.SampleLognormal(21, 61.4, 2) / 60#
      s.EntityAttribute(newEntity, s.SymbolNumber("atTlimite")) = s.RunCurrentTime + 180
    ElseIf i = 9 Then
      s.EntityAttribute(newEntity, s.SymbolNumber("atTtrabajo")) = s.SampleLognormal(6.2, 0.79, 2) / 60#
      s.EntityAttribute(newEntity, s.SymbolNumber("atTlimite")) = s.RunCurrentTime + 10080
    Else
      s.EntityAttribute(newEntity, s.SymbolNumber("atTtrabajo")) = s.SampleLognormal(13.11, 4.63, 2) / 60#
      s.EntityAttribute(newEntity, s.SymbolNumber("atTlimite")) = s.RunCurrentTime + 1440
    End If
    '... Enviamos el trabajo a la estacion de entrada de los servidores
    s.EntitySendToStation newEntity, estacionColaPrincipal
  End If
Next i

End Sub

Sub OLT1_B1F()

  '* Algoritmo EDF - Earliest Deadline First
  '* Se identifica la tarea que tiene el Tlimite más abajo y se asigna al servidor que
  '* más capacidad disponible tenga en ese momento
  '* El algoritmo distribuye tareas mientras hayan tareas en cola o procesadores disponibles
  '* Sólo se considera una cola principal (Global Scheduling Algorithm)

  Dim m As Model
```

```
Dim s As Arena.SIMAN
Dim numberInColaPrincipal As Integer, i As Integer
Dim EntidadSeleccionada As Long, estacionServidores As Long, res As Long
Dim varNumservidores As Integer, j As Integer
Dim ServidorNumber(10) As Long, numCoresOcupados(10) As Integer, numCoresLibres(10) As Integer
Dim ColaPrincipalNumber As Long
Dim ocupacionLibre As Integer, ServidorEscogido As Integer, freecores As Integer
Dim atTrabajoId As Long, atValue As Long
Dim atTlimite As Long

Dim Nollega As Boolean

Set m = ThisDocument.Model
Set s = m.SIMAN

'... El numero de servidores se fija desde Arena
varNumservidores = s.VariableArrayValue(s.SymbolNumber("varNumServidores"))
'... Inicializo el contador de cores libres
freecores = 0
'... Reviso la capacidad libre de cada servidor y la sumo en "freecores"
For i = 1 To varNumservidores
  '... puntero al servidor
  ServidorNumber(i) = s.SymbolNumber("RServidor" & i)
  '... capacidad utilizada de cada servidor
  numCoresOcupados(i) = s.ResourceNumberBusy(ServidorNumber(i))
  '... capacidad libre de cada servidor
  numCoresLibres(i) = s.ResourceCapacity(ServidorNumber(i)) - numCoresOcupados(i)
  freecores = freecores + numCoresLibres(i)
Next i

'... Enviamos tareas a todos los cores libres que haya, mientras tenga tareas en cola (Cola Principal)
ColaPrincipalNumber = s.SymbolNumber("Cola Principal.Queue")
numberInColaPrincipal = s.QueueNumberOfEntities(ColaPrincipalNumber)
While ((freecores > 0) And (numberInColaPrincipal > 0))

   '... Busco el servidor que más procesadores libres tiene disponibles
   ServidorEscogido = 1
   ocupacionLibre = numCoresLibres(ServidorEscogido)

   For j = 1 To varNumservidores
      If ocupacionLibre < numCoresLibres(j) Then
         ServidorEscogido = j
         ocupacionLibre = numCoresLibres(ServidorEscogido)
      End If
   Next j

   'Si alguno de los servidores tiene procesadores libres, le enviamos el trabajo más prioritario
   If ocupacionLibre > 0 Then
      '... Busco y selecciono la tarea más prioritaria para enviar al servidor con procesador libre
      Call LocalizaEDFTask(EntidadSeleccionada, ColaPrincipalNumber)
      '... Quitamos la tarea seleccionada de la cola principal
      s.QueueRemoveEntity EntidadSeleccionada, ColaPrincipalNumber
      '... Asignamos el trabajo al servidor escogido
      s.EntityAttribute(EntidadSeleccionada, s.SymbolNumber("atNumServidor")) = ServidorEscogido
      '... Enviamos el trabajo a la estacion de entrada de los servidores en caso de que se pueda acabar a tiempo
      '... Antes, verificamos si la tarea que se asigna se puede atender con tiempo suficente
      estacionServidores = s.SymbolNumber("Station Cola Servidores")
      Nollega = False
      Call CompruebaSiLlegaP(EntidadSeleccionada, Nollega)
```

```
        If Nollega = False Then
          s.EntitySendToStation EntidadSeleccionada, estacionServidores
          freecores = freecores - 1
        End If
      End If
      '... Actualizamos el número de trabajos en cola principal antes de volver a realizar un nuevo loop
      numberInColaPrincipal = s.QueueNumberOfEntities(ColaPrincipalNumber)
    Wend
End Sub

Sub OLT1_B3F()

  '* Evento asociado a la llegada de una entidad a la cola local.
  '* Se coge el trabajo y se envia al servidor escogido en el bloque anterior
  '* No se implementa cola local real.

  Dim m As Model
  Dim s As Arena.SIMAN
  Dim NumberInColaServidor As Integer, i As Integer
  Dim treball As Long, estacionServidores As Long
  Dim numServidores As Integer, j As Integer
  Dim ServidorNumber As Long, ColaServidorNumber As Long
  Dim atNumServidor As Integer
  Dim atTrabajoId As Long, atValue As Long
  Dim Nollega As Boolean

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  '... Id del trabajo
  atTrabajoId = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("atTrabajoId"))
  '... numero de servidor
  atNumServidor = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("atNumServidor"))
  '... puntero al servidor
  ServidorNumber = s.SymbolNumber("RServidor" & atNumServidor)
  '... puntero a la cola local
  ColaServidorNumber = s.SymbolNumber("ColaServidor" & atNumServidor)

  '... El nombre de la estacion es el nombre que aparece en el campo "Station name"
  estacionServidores = s.SymbolNumber("Station Servidores")

  '... Quitamos los trabajos de la cola local y los enviamos al servidor
  NumberInColaServidor = s.QueueNumberOfEntities(ColaServidorNumber)
  For i = 1 To NumberInColaServidor
    '... leemos el valor del atributo atTrabajoId de la entidad en la posicion i de la cola
    atValue = s.QueuedEntityAttribute(ColaServidorNumber, i, s.SymbolNumber("atTrabajoId"))
    If atValue = atTrabajoId Then
      '... Buscamos el trabajo que esta en la posición i de la cola
      treball = s.QueueEntityLocationAtRank(i, ColaServidorNumber)
      '... Quitamos el trabajo de la cola local
      s.QueueRemoveEntity treball, ColaServidorNumber
      '... Asigno valor de instante de tiempo actual a 'atTstart'
      '... para cálculos de tiempos posteriores estadísticos y de decisión
      s.EntityAttribute(treball, s.SymbolNumber("atTstart")) = s.RunCurrentTime
      '... Enviamos el trabajo al servidor en caso de que se pueda acabar a tiempo
      '... Antes, verificamos si la tarea que se asigna se puede atender con tiempo suficente
      Call CompruebaSiLlegaC(treball, Nollega, atNumServidor)
      If Nollega = False Then
```

```
            '... Enviamos el trabajo a la estación de entrada de los servidores
            s.EntitySendToStation treball, estacionServidores
        End If
    End If
Next i

End Sub

Private Sub VBA_Block_4_Fire()

  '... Evento asociado al fin de proceso de un trabajo para actualizar estadísticas acumuladas
  '...

  Dim m As Model
  Dim s As Arena.SIMAN
  Dim atNumServidor As Integer, atTipoTrabajo As Integer
  Dim SumLead As Double, SumLead2 As Double, atLead As Double, atLead2 As Double
  Dim varNumTrabajos As Integer

  Dim media As Double, value As Double
  Dim TSumAvg As Double, TSumAvg2 As Double, atTavg As Double, atTavg2 As Double
  Dim desviaciontipo As Double, atHalfWidth As Double

  Dim NumProcessed As Integer
  Dim NumLate As Integer, NumOnTime As Integer, ImpossibleOnTime As Integer
  Dim EarlinessAvg As Double, LatenessAvg As Double
  Dim EarlyLate As Double

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  '... Asignamos el nº de tipos de trabajos que tenemos definido
  varNumTrabajos = s.VariableArrayValue(s.SymbolNumber("varNumTrabajos"))

  '=== ACTUALIZACION DE ESTADISTICAS
  '... La primera acción es actualizar la cola de trabajos para los calculos estadisticos
  atNumServidor = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("atNumServidor"))
  atTipoTrabajo = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("atTipoTrabajo"))
  '.....
  '... AVERAGE TIME -- Tiempos medios de proceso
  '... Sumar a la suma de tiempos en el sistema
  TSumAvg = s.VariableArrayValue(s.SymbolNumber("varSumAvg", atNumServidor, atTipoTrabajo))
  atTavg = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("atTavg"))
  TSumAvg = TSumAvg + atTavg
  s.VariableArrayValue(s.SymbolNumber("varSumAvg", atNumServidor, atTipoTrabajo)) = TSumAvg
  '... Sumar a la suma de tiempos al cuadrado en el sistema
  TSumAvg2 = s.VariableArrayValue(s.SymbolNumber("varSumAvg2", atNumServidor, atTipoTrabajo))
  atTavg2 = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("atTavg2"))
  TSumAvg2 = TSumAvg2 + atTavg2
  s.VariableArrayValue(s.SymbolNumber("varSumAvg2", atNumServidor, atTipoTrabajo)) = TSumAvg2
  '... Actualizar el nº de trabajos procesados para su tipo de trabajo
  NumProcessed = s.VariableArrayValue(s.SymbolNumber("varNumProcessed", atNumServidor, atTipoTrabajo)) + 1
  s.VariableArrayValue(s.SymbolNumber("varNumProcessed", atNumServidor, atTipoTrabajo)) = NumProcessed

  '... La segunda acción es el re-cálculo de la media y del intervalo de confianza y almacenamiento
  '    de los nuevos valores en las matrices correspondientes

  '... Media
  media = TSumAvg / NumProcessed
```

```
s.VariableArrayValue(s.SymbolNumber("varTavg", atNumServidor, atTipoTrabajo)) = media
'... Desviación tipo
value = (TSumAvg2 / NumProcessed) - (media * media)
If value <= 0 Then
  desviaciontipo = 0
Else
  desviaciontipo = Sqr(value)
End If

'... Half Width (99.9% de nivel de confianza para 200 muestras)
atHalfWidth = 3.339 * desviaciontipo / Sqr(NumProcessed)
s.VariableArrayValue(s.SymbolNumber("varDesviacionAvg", atNumServidor, atTipoTrabajo)) = desviaciontipo
s.VariableArrayValue(s.SymbolNumber("varHalfwidthAvg", atNumServidor, atTipoTrabajo)) = atHalfWidth

'... Valoración de los datos de Earliness & Lateness para cada tipo de trabajo

'... El tiempo sobrante se toma de la diferencia entre el Deadline solicitado y el instante de
'... tiempo en que finaliza el procesamiento del trabajo. El dato viene de Arena
EarlyLate = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("atTSobrante"))

If EarlyLate < 0 Then
    '... El trabajo se ha entregado fuera del plazo máximo
    NumLate = s.VariableArrayValue(s.SymbolNumber("varNumLate", atNumServidor, atTipoTrabajo))
    LatenessAvg = s.VariableArrayValue(s.SymbolNumber("varLatenessAvg", atNumServidor, atTipoTrabajo))
    LatenessAvg = ((NumLate * LatenessAvg) - EarlyLate) / (NumLate + 1)
    NumLate = NumLate + 1
    s.VariableArrayValue(s.SymbolNumber("varLatenessAvg", atNumServidor, atTipoTrabajo)) = LatenessAvg
    s.VariableArrayValue(s.SymbolNumber("varNumLate", atNumServidor, atTipoTrabajo)) = NumLate

Else
    '... El trabajo ha sido entregado dentro del plazo máximo
    NumOnTime = s.VariableArrayValue(s.SymbolNumber("varNumOnTime", atNumServidor, atTipoTrabajo))
    EarlinessAvg = s.VariableArrayValue(s.SymbolNumber("varEarlinessAvg", atNumServidor, atTipoTrabajo))
    EarlinessAvg = ((NumOnTime * EarlinessAvg) + EarlyLate) / (NumOnTime + 1)
    NumOnTime = NumOnTime + 1
    s.VariableArrayValue(s.SymbolNumber("varEarlinessAvg", atNumServidor, atTipoTrabajo)) = EarlinessAvg
    s.VariableArrayValue(s.SymbolNumber("varNumOnTime", atNumServidor, atTipoTrabajo)) = NumOnTime
End If

'... Registro de la ocupación de cálculo generada en los servidores
'... Cada hora de simulación, se guarda la ocupación de cada servidor en
'... la variable 'Utilización(i)' para su volcado horario a fichero externo

Dim varNumservidores As Integer, varNumCores As Integer, varHora As Integer
Dim UsoHora As Integer, Tstart As Double, Tfin As Double, resto As Double
Dim UsoHorario As Double, UsohorarioSig As Double, UsoHorarioAnt As Double
Dim i As Integer

varNumservidores = s.VariableArrayValue(s.SymbolNumber("varNumServidores"))
varNumCores = s.VariableArrayValue(s.SymbolNumber("varNumCores"))
varHora = s.VariableArrayValue(s.SymbolNumber("varHora"))
UsoHora = s.VariableArrayValue(s.SymbolNumber("varUsoHora"))
Tstart = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("atTstart"))
Tfin = Tstart + atTavg

If (UsoHora Mod 24 < varHora) Or ((varHora = 0) And (UsoHora Mod 24 = 23)) Then
    UsoHora = UsoHora + 1
    s.VariableArrayValue(s.SymbolNumber("varUsoHora")) = UsoHora
    For i = 1 To varNumservidores
```

```
        UsoHorarioAnt = s.VariableArrayValue(s.SymbolNumber("VarOcupServ", i, 1))
        UsoHorario = s.VariableArrayValue(s.SymbolNumber("VarOcupServ", i, 2))
        UsohorarioSig = s.VariableArrayValue(s.SymbolNumber("VarOcupServ", i, 3))
        s.VariableArrayValue(s.SymbolNumber("Utilizacion", i)) = UsoHorarioAnt / (varNumCores * 60)
        s.VariableArrayValue(s.SymbolNumber("VarOcupServ", i, 1)) = UsoHorario
        s.VariableArrayValue(s.SymbolNumber("VarOcupServ", i, 2)) = UsohorarioSig
        s.VariableArrayValue(s.SymbolNumber("VarOcupServ", i, 3)) = 0
    Next i
End If

UsoHorarioAnt = s.VariableArrayValue(s.SymbolNumber("VarOcupServ", atNumServidor, 1))
UsoHorario = s.VariableArrayValue(s.SymbolNumber("VarOcupServ", atNumServidor, 2))
UsohorarioSig = s.VariableArrayValue(s.SymbolNumber("VarOcupServ", atNumServidor, 3))

Select Case UsoHora
    Case 0

    Case Else
        If Tstart < 60 * (UsoHora - 1) Then
            resto = Tstart Mod 60
            UsoHorarioAnt = UsoHorarioAnt + 60 - resto
            UsoHorario = UsoHorario + atTavg - 60 + resto
            If Tfin > 60 * UsoHora Then
                resto = Tfin Mod 60
                UsoHorario = UsoHorario - resto
                UsohorarioSig = UsohorarioSig + resto
            End If
            GoTo Finalizar
        End If
        If Tstart < 60 * UsoHora Then
            UsoHorario = UsoHorario + atTavg
            If Tfin > 60 * UsoHora Then
                resto = Tfin Mod 60
                UsoHorario = UsoHorario - resto
                UsohorarioSig = UsohorarioSig + resto
            End If
        Else
            UsohorarioSig = UsohorarioSig + atTavg
        End If
        s.VariableArrayValue(s.SymbolNumber("VarOcupServ", atNumServidor, 1)) = UsoHorarioAnt
        s.VariableArrayValue(s.SymbolNumber("VarOcupServ", atNumServidor, 2)) = UsoHorario
        s.VariableArrayValue(s.SymbolNumber("VarOcupServ", atNumServidor, 3)) = UsohorarioSig
  End Select
Finalizar:
End Sub
```

PROCEDIMIENTOS AUXILIARES:

Los procedimientos principales anteriores hacen uso de unos procedimientos auxiliares que se exponen a continuación. Las funciones de estos procedimientos auxiliares son:

1. Localizar la tarea que tenga mayor prioridad según el criterio EDF, en cada momento.

2. Estimar si un trabajo va a ser entregado a tiempo, antes de enviarlo a cola local.
3. Estimar si un trabajo va a ser entregado a tiempo antes de enviarlo al servidor.

```
Sub LocalizaEDFTask(EntidadSeleccionada As Long, ColaNumber As Long)

  '... Identifica el trabajo más prioritario (el de menor Tlimite) para la cola 'Colanumber'
  '... y lo devuelve a través de 'EntidadSeleccionada'

  Dim m As Model
  Dim s As Arena.SIMAN

  Dim atTlimiteMenor As Long, NumberInCola As Integer
  Dim atTlimite As Long
  Dim i As Integer

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  '... Inicializa los valores de atTlimiteMenor, EntidadSeleccionada y NumberInCola
  '... antes de iniciar la búsqueda del trabajo más prioritario
  atTlimiteMenor = s.QueuedEntityAttribute(ColaNumber, 1, s.SymbolNumber("atTlimite"))
  EntidadSeleccionada = s.QueueEntityLocationAtRank(1, ColaNumber)
  NumberInCola = s.QueueNumberOfEntities(ColaNumber)

  '... Buscamos la tarea más prioritaria a través de todos los trabajos presentes en cola
  For i = 1 To NumberInCola
     atTlimite = s.QueuedEntityAttribute(ColaNumber, i, s.SymbolNumber("atTlimite"))
     If atTlimite < atTlimiteMenor Then
        atTlimiteMenor = atTlimite
        EntidadSeleccionada = s.QueueEntityLocationAtRank(i, ColaNumber)
     End If
   Next i

End Sub

Sub CompruebaSiLlegaP(Entidad As Long, NollegaP As Boolean)

  '... Recibe un trabajo y comprueba si se espera cumplir el plazo
  '... con los datos disponibles hasta el momento
  '... Devolverá el resultado en el booleano NollegaP

  Dim m As Model
  Dim s As Arena.SIMAN

  Dim atNumServidores As Integer, NumProcessed As Integer, atTipoTrabajo As Integer
  Dim k As Integer, ServidorReferencia As Integer, DuracionEstimada As Double
  Dim ImpossibleOnTime As Integer

  Set m = ThisDocument.Model
  Set s = m.SIMAN

   '... Identificamos el número de Servidores y el Tipo de Trabajo
   '... del trabajo que estamos verificando
   atNumServidores = s.VariableArrayValue(s.SymbolNumber("varNumServidores"))
   atTipoTrabajo = s.EntityAttribute(Entidad, s.SymbolNumber("atTipoTrabajo"))

   '...Identificamos el servidor que tiene más datos disponibles para el tipo de trabajo en cuestión
```

```
  ServidorReferencia = 1
  NumProcessed = s.VariableArrayValue(s.SymbolNumber("varNumProcessed", ServidorReferencia, atTipoTrabajo))

  For k = 1 To atNumServidores
     If NumProcessed < s.VariableArrayValue(s.SymbolNumber("varNumProcessed", k, atTipoTrabajo)) Then
     NumProcessed = s.VariableArrayValue(s.SymbolNumber("varNumProcessed", k, atTipoTrabajo))
     ServidorReferencia = k
     End If
  Next k

  '... Compruebo si el trabajo puede ser entregado a tiempo en función de los datos disponibles
  '... en caso de considerar que disponemos de la cantidad de datos mínima necesaria
  If NumProcessed > 15 Then ' Este valor se puede cambiar en función del grado de precisión que queremos
  '... Tomamos el valor estimado de la duración del trabajo del histórico de duraciones para el mismo
  '... Tipo de Trabajo
  DuracionEstimada = s.VariableArrayValue(s.SymbolNumber("varTavg", ServidorReferencia, atTipoTrabajo))
     'Verificamos si el Trabajo puede NO ser entregado dentro de los plazos solicitados
     If (s.EntityAttribute(Entidad, s.SymbolNumber("atTlimite")) - s.RunCurrentTime) < DuracionEstimada Then
        NollegaP = True
        '... Acumulamos el nº de trabajos expulsados antes de salir de la Cola Principal
        ImpossibleOnTime = s.VariableArrayValue(s.SymbolNumber("varImpossibleOnTime", 3, atTipoTrabajo))
        ImpossibleOnTime = ImpossibleOnTime + 1
        s.VariableArrayValue(s.SymbolNumber("varImpossibleOnTime", 3, atTipoTrabajo)) = ImpossibleOnTime
     End If
  '... Guardamos el valor de DuracionEstimada para el Trabajo en el atributo 'atTestimado'
  s.EntityAttribute(Entidad, s.SymbolNumber("atTestimado")) = DuracionEstimada
  End If
End Sub

Sub CompruebaSiLlegaC(Entidad As Long, Nollega As Boolean, ServidorEscogido As Integer)

  '... Recibe un trabajo y comprueba si se espera cumplir el plazo con los datos
  '... disponibles hasta el momento en el servidor en que se va a ejecutar
  '... Devolverá el resultado en el booleano Nollega

  Dim m As Model
  Dim s As Arena.SIMAN

  Dim atNumServidores As Integer, NumProcessed As Integer, atTipoTrabajo As Integer
  Dim k As Integer, ServidorReferencia As Integer, DuracionEstimada As Double
  Dim ImpossibleOnTime As Integer

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  '... Identificamos el número de Servidores y el Tipo de Trabajo
  '... del trabajo que estamos verificando
  atNumServidores = s.VariableArrayValue(s.SymbolNumber("varNumServidores"))
  atTipoTrabajo = s.EntityAttribute(Entidad, s.SymbolNumber("atTipoTrabajo"))

  '...Identificamos el servidor que tiene más datos disponibles para el tipo de trabajo en cuestión
  ServidorReferencia = 1
  NumProcessed = s.VariableArrayValue(s.SymbolNumber("varNumProcessed", ServidorReferencia, atTipoTrabajo))

  For k = 1 To atNumServidores
     If NumProcessed < s.VariableArrayValue(s.SymbolNumber("varNumProcessed", k, atTipoTrabajo)) Then
     NumProcessed = s.VariableArrayValue(s.SymbolNumber("varNumProcessed", k, atTipoTrabajo))
     ServidorReferencia = k
     End If
```

```
  Next k

  '... Compruebo si el trabajo puede ser entregado a tiempo en función de los datos disponibles
  '... en caso de considerar que disponemos de la cantidad de datos mínima necesaria
  If NumProcessed > 15 Then ' Este valor se puede cambiar en función del grado de precisión que queremos
  DuracionEstimada = s.VariableArrayValue(s.SymbolNumber("varTavg", ServidorEscogido, atTipoTrabajo))
    If (s.EntityAttribute(Entidad, s.SymbolNumber("atTlimite")) - s.EntityAttribute(Entidad,
s.SymbolNumber("atTStart"))) < DuracionEstimada Then
        Nollega = True
        '... Acumulamos el nº de trabajos expulsados antes de salir de la Cola Local
        ImpossibleOnTime = s.VariableArrayValue(s.SymbolNumber("varImpossibleOnTime", ServidorEscogido,
atTipoTrabajo))
        ImpossibleOnTime = ImpossibleOnTime + 1
        s.VariableArrayValue(s.SymbolNumber("varImpossibleOnTime", ServidorEscogido, atTipoTrabajo)) =
ImpossibleOnTime
    End If
  End If
End Sub
```

B.2: Algoritmo LLF

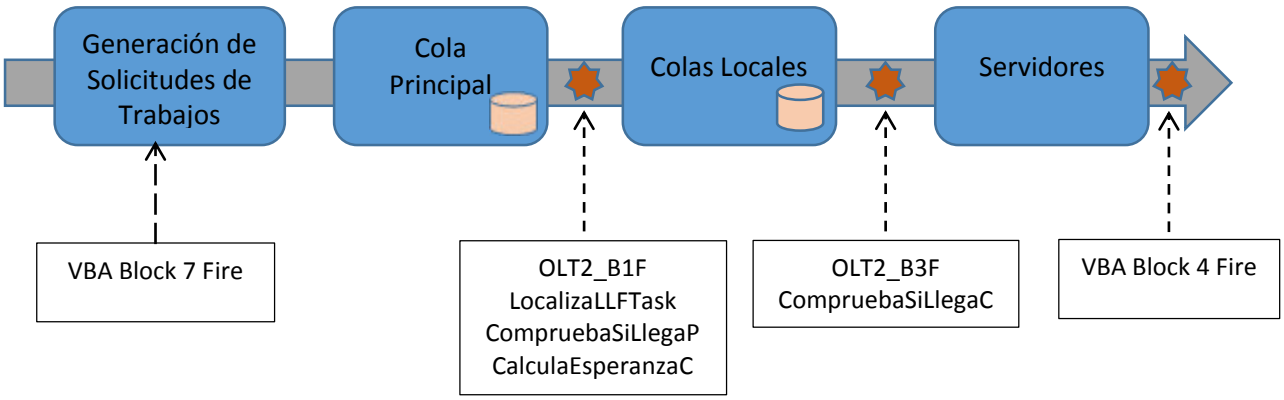


Gráfico B.2

Los procedimientos principales desencadenados por los eventos durante la simulación con planificación **LLF** son los siguientes:

Private Sub **VBA_Block_7_Fire()**

Corresponde al mismo procedimiento utilizado en el caso de planificación EDF. Su código puede encontrarse en el apartado B.1 de este anexo.

Private Sub **VBA_Block_4_Fire()**

Corresponde al mismo procedimiento utilizado en el caso de planificación EDF. Su código puede encontrarse en el apartado B.1 de este anexo.

Sub **OLT2_B1F()**

```
'* Algoritmo LLF - Least Laxity First
'* Se identifica la tarea que tiene el margen de comienzo menor (Holgura)
'* y se asigna al servidor que más capacidad disponible tenga
'* El algoritmo distribuye tareas mientras hayan tareas en cola o procesadores disponibles
'* Sólo se considera una cola principal (Global Scheduling Algorithm)

Dim m As Model
Dim s As Arena.SIMAN
Dim numberInColaPrincipal As Integer, i As Integer
Dim EntidadSeleccionada As Long, estacionServidores As Long, res As Long
Dim EntidadIntermedia As Long
Dim varNumservidores As Integer, j As Integer
Dim ServidorNumber(10) As Long, numCoresOcupados(10) As Integer, numCoresLibres(10) As Integer
Dim ColaPrincipalNumber As Long
Dim ocupacionLibre As Integer, ServidorEscogido As Integer, freecores As Integer
Dim atTrabajoId As Long, atValue As Long
Dim atTlimite As Long, atTlimiteMenor As Long, atTtrabajo As Double
Dim LLF As Double, atLLFMenor As Double

Dim Nollega As Boolean

Set m = ThisDocument.Model
Set s = m.SIMAN

'... El numero de servidores se fija desde Arena
varNumservidores = s.VariableArrayValue(s.SymbolNumber("varNumServidores"))
'... Inicializo el contador de cores libres
freecores = 0
'... Reviso la capacidad libre de cada servidor y la sumo en "freecores"
For i = 1 To varNumservidores
  '... puntero al servidor
  ServidorNumber(i) = s.SymbolNumber("RServidor" & i)
  '... capacidad utilizada de cada servidor
  numCoresOcupados(i) = s.ResourceNumberBusy(ServidorNumber(i))
  '... capacidad libre de cada servidor
  numCoresLibres(i) = s.ResourceCapacity(ServidorNumber(i)) - numCoresOcupados(i)
  freecores = freecores + numCoresLibres(i)
Next i

'... Enviamos tareas a todos los cores libres que haya, mientras tenga tareas en cola (Cola Principal)
ColaPrincipalNumber = s.SymbolNumber("Cola Principal.Queue")
numberInColaPrincipal = s.QueueNumberOfEntities(ColaPrincipalNumber)
While ((freecores > 0) And (numberInColaPrincipal > 0))

    '... Busco el servidor que más procesadores libres tiene disponibles
    ServidorEscogido = 1
    ocupacionLibre = numCoresLibres(ServidorEscogido)
    For j = 1 To varNumservidores
       If ocupacionLibre < numCoresLibres(j) Then
          ServidorEscogido = j
          ocupacionLibre = numCoresLibres(ServidorEscogido)
       End If
    Next j

    'Si alguno de los servidores tiene procesadores libres, le enviamos el trabajo más prioritario
    If ocupacionLibre > 0 Then
    '... Busco y selecciono la tarea más prioritaria para enviar al servidor con procesador libre
       Call LocalizaLLFTask(EntidadSeleccionada, ColaPrincipalNumber, ServidorEscogido)
       '... Quitamos la tarea seleccionada de la cola principal
```

```
        s.QueueRemoveEntity EntidadSeleccionada, ColaPrincipalNumber
        '... Asignamos el trabajo al servidor escogido
        s.EntityAttribute(EntidadSeleccionada, s.SymbolNumber("atNumServidor")) = ServidorEscogido
        '... Enviamos el trabajo a la estacion de entrada de los servidores en caso de que se pueda acabar a tiempo
        '... Antes, verificamos si la tarea que se asigna se puede atender con tiempo suficente
        estacionServidores = s.SymbolNumber("Station Cola Servidores")
        Nollega = False
        Call CompruebaSiLlegaP(EntidadSeleccionada, Nollega)
        If Nollega = False Then
          s.EntitySendToStation EntidadSeleccionada, estacionServidores
          freecores = freecores - 1
        End If
      End If
      '... Actualizamos el número de trabajos en cola principal antes de volver a realizar un nuevo loop
      numberInColaPrincipal = s.QueueNumberOfEntities(ColaPrincipalNumber)
  Wend
End Sub


Sub OLT2_B3F()

  '* Evento asociado a la llegada de una entidad a la cola local.
  '* Se coge el trabajo y se envia al servidor escogido en el bloque anterior
  '* No se implementa cola local real.

  Dim m As Model
  Dim s As Arena.SIMAN
  Dim NumberInColaServidor As Integer, i As Integer
  Dim numServidores As Integer, j As Integer
  Dim treball As Long, estacionServidores As Long
  Dim ServidorNumber As Long, ColaServidorNumber As Long, atNumServidor As Integer
  Dim Nollega As Boolean
  Dim atTrabajoId As Long, atValue As Long

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  '... Id del trabajo
  atTrabajoId = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("atTrabajoId"))
  '... numero de servidor
  atNumServidor = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("atNumServidor"))
  '... puntero al servidor
  ServidorNumber = s.SymbolNumber("RServidor" & atNumServidor)
  '... puntero a la cola local
  ColaServidorNumber = s.SymbolNumber("ColaServidor" & atNumServidor)

  '... El nombre de la estacion es el nombre que aparece en el campo "Station name"
  estacionServidores = s.SymbolNumber("Station Servidores")

  '... Quitamos los trabajos de la cola local y los enviamos al servidor
  NumberInColaServidor = s.QueueNumberOfEntities(ColaServidorNumber)
  For i = 1 To NumberInColaServidor
    '... leemos el valor del atributo atTrabajoId de la entidad en la posicion i de la cola
    atValue = s.QueuedEntityAttribute(ColaServidorNumber, i, s.SymbolNumber("atTrabajoId"))
    If atValue = atTrabajoId Then
      '... Buscamos el trabajo que esta en la posición i de la cola
      treball = s.QueueEntityLocationAtRank(i, ColaServidorNumber)
      '... Quitamos el trabajo de la cola local
      s.QueueRemoveEntity treball, ColaServidorNumber
```

```
        '... Asigno valor de instante de tiempo actual a 'atTstart'
        '... para cálculos de tiempos posteriores estadísticos y de decisión
        s.EntityAttribute(treball, s.SymbolNumber("atTstart")) = s.RunCurrentTime
        '... Enviamos el trabajo al servidor en caso de que se pueda acabar a tiempo
        '... Antes, verificamos si la tarea que se asigna se puede atender con tiempo suficente
        Call CompruebaSiLlegaC(treball, Nollega, atNumServidor)
        If Nollega = False Then
            '... Enviamos el trabajo a la estacion de entrada de los servidores
            s.EntitySendToStation treball, estacionServidores
        End If
      End If
  Next i

End Sub
```

PROCEDIMIENTOS AUXILIARES:

Los procedimientos principales anteriores para planificación **LLF** hacen uso de unos procedimientos auxiliares que se exponen a continuación. Las funciones de estos procedimientos auxiliares son:

1. Localizar la tarea que tenga mayor prioridad según el criterio LLF, en cada momento.
2. Estimar la duración de un trabajo en un servidor definido.
3. Estimar si un trabajo va a ser entregado a tiempo, antes de enviarlo a cola local.
4. Estimar si un trabajo va a ser entregado a tiempo antes de enviarlo al servidor.

```
Sub LocalizaLLFTask(EntidadSeleccionada As Long, ColaNumber As Long, Servidor As Integer)

  '... Identifica el trabajo más prioritario (el de menor Holgura) para la cola 'Colanumber'
  '... y lo devuelve a través de 'EntidadSeleccionada'

  Dim m As Model
  Dim s As Arena.SIMAN

  Dim atLLFMenor As Long, atTlimite As Long, DuracionEstimada As Double
  Dim LLF As Long, EntidadIntermedia As Long
  Dim i As Integer, NumberInCola As Integer

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  '... Inicializa los valores de atTlimite, EntidadSeleccionada y NumberInCola
  '... antes de iniciar la búsqueda del trabajo más prioritario
  i = 1
  EntidadSeleccionada = s.QueueEntityLocationAtRank(i, ColaNumber)
  NumberInCola = s.QueueNumberOfEntities(ColaNumber)
  atTlimite = s.QueuedEntityAttribute(ColaNumber, i, s.SymbolNumber("atTlimite"))
  '... Para inicializar la variable 'atLLFMenor' debemos antes calcular la estimación de la
  '... duración prevista para el trabajo que se está valorando con 'CalculaEsperanzaC'
  Call CalculaEsperanzaC(EntidadSeleccionada, Servidor, DuracionEstimada)
  atLLFMenor = atTlimite - DuracionEstimada
  '... Si la 'DuracionEstimada' del trabajo es 0, lo envía directamente al servidor para
```

```
'... su ejecución y poder disponer de datos estadísticos futuros
'... En caso contrario, se sigue buscando el trabajo en cola que tenga la Holgura menor
While (DuracionEstimada <> 0) And (i <= NumberInCola)
    atTlimite = s.QueuedEntityAttribute(ColaNumber, i, s.SymbolNumber("atTlimite"))
    EntidadIntermedia = s.QueueEntityLocationAtRank(i, ColaNumber)
    Call CalculaEsperanzaC(EntidadIntermedia, Servidor, DuracionEstimada)
    LLF = atTlimite - DuracionEstimada
    '... Si la holgura es menor que 'atLLFMenor' se actualizan los valores de
    '... 'atLLFMenor' y 'EntidadSeleccionada'
    If LLF < atLLFMenor Then
       atLLFMenor = LLF
       EntidadSeleccionada = EntidadIntermedia
    End If
    '... Si la 'DuracionEstimada' es 0, se actualiza el valor de 'EntidadSeleccionada'
    '... y se saldrá del bucle, con esa entidad como prioritaria
    If DuracionEstimada = 0 Then
       EntidadSeleccionada = EntidadIntermedia
    End If
    i = i + 1
 Wend

End Sub

Sub CalculaEsperanzaC(Entidad As Long, Servidor As Integer, DuracionEstimada As Double)

  '... Recibe un Trabajo y el Servidor en que se va a ejecutar para devolver el valor
  '... (o Esperanza) de la duración estimada en base a los datos históricos disponibles
  '... para cada tipo de trabajo

  Dim m As Model
  Dim s As Arena.SIMAN

  Dim atNumServidor As Integer, NumProcessed As Integer, atTipoTrabajo As Integer
  Dim k As Integer, ServidorReferencia As Integer
  Dim ImpossibleOnTime As Integer

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  '... Leemos el tipo al que pertenece el trabajo recibido
  atTipoTrabajo = s.EntityAttribute(Entidad, s.SymbolNumber("atTipoTrabajo"))

  '...Buscamos el valor medio de duración del tipo de trabajo en el servidor que se ha asignado
  DuracionEstimada = s.VariableArrayValue(s.SymbolNumber("varTavg", Servidor, atTipoTrabajo

End Sub
```

Sub **CompruebaSiLlegaP(Entidad As Long, NollegaP As Boolean)**

Corresponde al mismo procedimiento utilizado en el caso de planificación EDF. Su código puede encontrarse en el apartado B.1 de este anexo.

Sub **CompruebaSiLlegaC(Entidad As Long, Nollega As Boolean, ServidorEscogido As Integer)**

Corresponde al mismo procedimiento utilizado en el caso de planificación EDF. Su código puede encontrarse en el apartado B.1 de este anexo.

B.3: Algoritmo MYOPIC

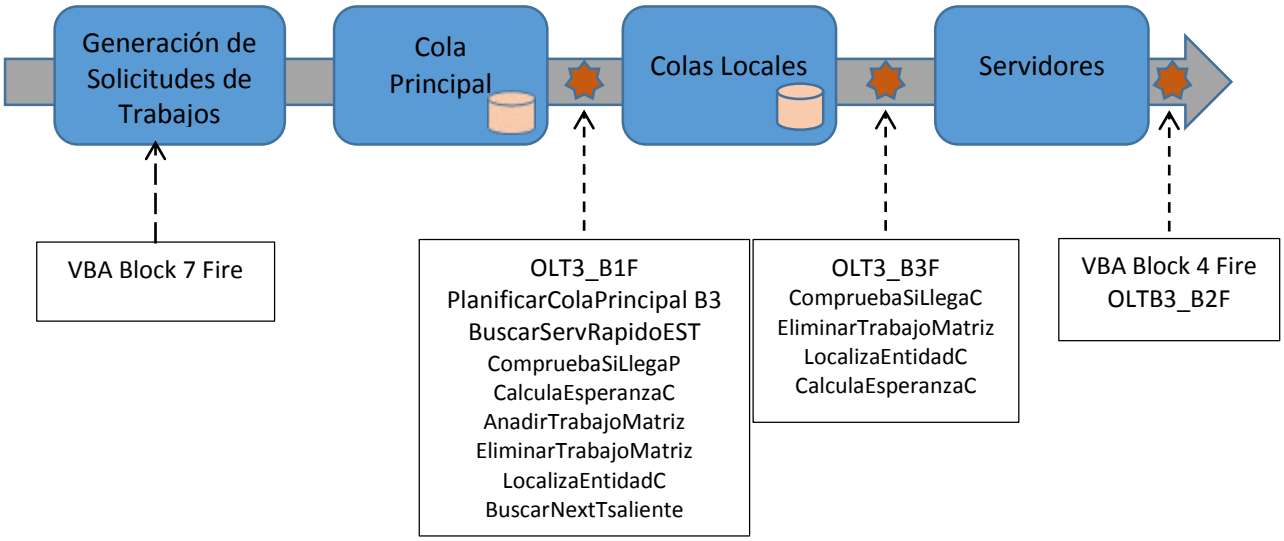


Gráfico B.3

Los procedimientos principales desencadenados por los eventos durante la simulación con planificación **MYOPIC** son los siguientes:

Private Sub **VBA_Block_7_Fire()**

Corresponde al mismo procedimiento utilizado en el caso de planificación EDF. Su código puede encontrarse en el apartado B.1 de este anexo.

Private Sub **VBA_Block_4_Fire()**

Corresponde al mismo procedimiento utilizado en el caso de planificación EDF. Su código puede encontrarse en el apartado B.1 de este anexo.

Sub **OLT3_B1F()**

```
'   ALGORITMO MYOPIC APLICADO A SERVIDORES DE n PROCESADORES CADA UNO
'   Esta rutina se ejecuta cada vez que llega una solicitud de trabajo nueva
'   a Cola Principal, para guardar la solicitud en una matriz ordenada por
'   prioridad EDF

Dim m As Model
Dim s As Arena.SIMAN
Dim numberInColaPrincipal As Integer, i As Integer
Dim varNumservidores As Integer, varNumCores As Integer
Dim ColaPrincipalNumber As Long

Dim LengthMatrizP As Integer
```

```
  Set m = ThisDocument.Model
  Set s = m.SIMAN

  '... Fijamos la cola de la que vamos a leer la solicitud y
  '... evaluamos la longitud de la misma
  ColaPrincipalNumber = s.SymbolNumber("Cola Principal.Queue")
  numberInColaPrincipal = s.QueueNumberOfEntities(ColaPrincipalNumber)
  '... Añadimos la solicitud de trabajo a la matriz de solicitudes ordenadas
  '... por prioridad EDF. En la matriz se guarda el atributo atTrabajoId de la solicitud
  LengthMatrizP = s.VariableArrayValue(s.SymbolNumber("varLengthMatrizP"))
  Call AnadirTrabajoMatriz(s.ActiveEntity, "VarMatrizColaP", LengthMatrizP, ColaPrincipalNumber,
numberInColaPrincipal, 1)
  s.VariableArrayValue(s.SymbolNumber("varLengthMatrizP")) = LengthMatrizP

  '... Lanzamos una planificación de las solicitudes acumuladas en Cola Principal
  Call PlanificarColaPrincipal_B3

End Sub

Sub PlanificarColaPrincipal_B3()

  '   Algoritmo principal de planificación Myopic
  '   Para cada solicitud presente en la matriz de solicitudes ordenadas por EDF
  '   se comprueba cuál de los servidores puede realizar el trabajo con una fecha
  '   de finalización prevista menor
  '   Antes de enviar un trabajo a cola de un servidor, se comprueba:
  '   1. Que el trabajo puede entregarse dentro del plazo establecido
  '   2. Que la planificación de la cola a la que va aser asignado es viable
  '
  '   Este algoritmo se ejecuta cada vez que llega una solicitud nueva a Cola Principal
  '   y cada vez que un procesador queda disponible
  '   También se ejecuta cada vez que hay capacidad disponible en alguno de los servidores
  '
  '   Hay una verificación adicional: Si un trabajo no va a poder ser entregado a tiempo,
  '   es eliminado de la Cola Principal para evitar enviar un trabajo a un procesador
  '   sabiendo que no se va a cumplir su plazo solicitado

  Dim m As Model
  Dim s As Arena.SIMAN
  Dim numberInColaPrincipal As Integer, NumberInServidor As Integer, i As Integer
  Dim Entidad As Long, estacionServidores As Long
  Dim Numcores As Integer, ServidorEscogido As Integer
  Dim ColaPrincipalNumber As Long, ColaServidorNumber As Long
  Dim atTrabajoId As Long
  Dim varNumservidores As Integer, freecores As Integer, forecast As Integer
  Dim ServidorNumber(10) As Long, numCoresOcupados(10) As Integer, numCoresLibres(10) As Integer
  Dim Nollega As Boolean, NollegaP As Boolean
  Dim LengthMatrizP As Integer, LengthMatrizL As Integer
  Dim MaxColaServidor(10) As Integer, NumberInColaServidor(10) As Integer

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  '... Identificamos la Cola Principal, la longitud de la matriz de solicitudes ordenadas
  '... y la estación de servidores
  ColaPrincipalNumber = s.SymbolNumber("Cola Principal.Queue")
  LengthMatrizP = s.VariableArrayValue(s.SymbolNumber("varLengthMatrizP"))
  estacionServidores = s.SymbolNumber("Station Cola Servidores")
```

```
MaxColaServidor(1) = 7
MaxColaServidor(2) = 7
i = 1

'... El numero de servidores se fija desde Arena
varNumservidores = s.VariableArrayValue(s.SymbolNumber("varNumServidores"))
'... Inicializo el contador de procesadores libres
freecores = 0
'... Reviso la capacidad libre de cada servidor y la sumo en "freecores"
For i = 1 To varNumservidores
  '... puntero al servidor
  ServidorNumber(i) = s.SymbolNumber("RServidor" & i)
  '... Registramos la longitud de la cola en el Servidor i
  NumberInColaServidor(i) = s.QueueNumberOfEntities(ServidorNumber(i))
  '... capacidad utilizada de cada servidor
  numCoresOcupados(i) = s.ResourceNumberBusy(ServidorNumber(i))
  '... capacidad libre de cada servidor
  numCoresLibres(i) = s.ResourceCapacity(ServidorNumber(i)) - numCoresOcupados(i)
  freecores = freecores + numCoresLibres(i)
Next i

'... Cálculo del nº máximo de trabajos que se pueden poner en cola en los servidores
'... en función de los procesadores libres y de las longitudes de cola de cada servidor
forecast = freecores + MaxColaServidor(1) + MaxColaServidor(2)
forecast = forecast - NumberInColaServidor(1) - NumberInColaServidor(2)

'... Cargo con trabajos todos los servidores que haya y que tengan una planificación viable,
'... y mientras tenga tareas en cola

While i <= LengthMatrizP And i <= forecast
    atTrabajoId = s.VariableArrayValue(s.SymbolNumber("VarMatrizColaP", 1, i))
    '... Identificamos la solicitud buscando la entidad en la cola principal
    '... en base al atributo atTrabajoId
    Call LocalizarEntidadC(Entidad, atTrabajoId, ColaPrincipalNumber)
    '... Verificamos si la tarea que se quiere planificar se puede atender con tiempo suficente
    NollegaP = False
    Call CompruebaSiLlegaP(Entidad, NollegaP)
    If NollegaP = True Then
       '... Eliminamos la tarea i (atTrabjoId) de la cola principal y de la matriz de solicitudes ordenadas
       s.QueueRemoveEntity Entidad, ColaPrincipalNumber
       Call EliminarTrabajoMatriz(Entidad, "VarMatrizColaP", LengthMatrizP, 1)
       s.VariableArrayValue(s.SymbolNumber("varLengthMatrizP")) = LengthMatrizP
       GoTo SigTrabajo
    End If
    '... Buscamos el servidor que puede atender más rápido la solicitud i (o atTrabajoId)
    Call BuscarServRapidoEST(Entidad)
    If s.EntityAttribute(Entidad, s.SymbolNumber("atNumServidor")) <> 0 Then
       '... Se ha encontrado sitio en una cola del servidor 'atNumServidor'
       '... Se elimina la solicitud de la Cola Principal y de la matriz ordenada
       s.QueueRemoveEntity Entidad, ColaPrincipalNumber
       Call EliminarTrabajoMatriz(Entidad, "VarMatrizColaP", LengthMatrizP, 1)
       s.VariableArrayValue(s.SymbolNumber("varLengthMatrizP")) = LengthMatrizP
       '... Se lee el servidor asignado para el trabajo
       ServidorEscogido = s.EntityAttribute(Entidad, s.SymbolNumber("atNumServidor"))
       ColaServidorNumber = s.SymbolNumber("ColaServidor" & ServidorEscogido)
       NumberInServidor = s.QueueNumberOfEntities(ColaServidorNumber)
       '... Se incluye el trabajo en la matriz ordenada de la cola del servidor asignado
       LengthMatrizL = s.VariableArrayValue(s.SymbolNumber("varLengthMatrizL", ServidorEscogido, 1))
```

```
        Call AnadirTrabajoMatriz(Entidad, "VarMatrizColaL", LengthMatrizL, ColaServidorNumber,
NumberInServidor, ServidorEscogido)
        s.VariableArrayValue(s.SymbolNumber("varLengthMatrizL", ServidorEscogido, 1)) = LengthMatrizL
        '... Enviamos el trabajo a la cola del servidor seleccionado
        s.EntitySendToStation Entidad, estacionServidores
      End If
      i = i + 1
SigTrabajo:
  Wend
Call OLT3_B3F
End Sub

Sub OLT3_B3F()

  '   Evento asociado a la llegada de una entidad a la cola local.
  '   Si hay algún core libre, se envía el trabajo más prioritario
  '   según criterio EDF al servidor asignado durante la planificación previa

  Dim m As Model
  Dim s As Arena.SIMAN
  Dim NumberInColaServidor As Integer, varNumservidores As Integer, i As Integer
  Dim varNumCores As Integer, j As Integer, Tfck As Long
  Dim EntidadSeleccionada As Long, estacionServidores As Long
  Dim ServidorNumber As Long, atNumServidor As Integer
  Dim ColaServidorNumber As Long, atTrabajoId As Long
  Dim LengthMatrizL As Integer, CapacidadLibre As Integer
  Dim DuracionEstimada As Double, Nollega As Boolean

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  varNumCores = s.VariableArrayValue(s.SymbolNumber("varNumCores"))
  varNumservidores = s.VariableArrayValue(s.SymbolNumber("varNumServidores"))
  '... El nombre de la estacion es el nombre que aparece en el campo "Station name"
  estacionServidores = s.SymbolNumber("Station Servidores")
  '... Para cada servidor, se verifica si tienen algún procesador libre y, siempre
  '... que tenga trabajos en cola, se le envía el más prioritario según criterio EDF
  '... La selección del trabajo se hace con la matriz ordenada (primer elemento)del servidor concreto
  For i = 1 To varNumservidores
      ServidorNumber = s.SymbolNumber("RServidor" & i)
      CapacidadLibre = s.ResourceCapacity(ServidorNumber) - s.ResourceNumberBusy(ServidorNumber)
      LengthMatrizL = s.VariableArrayValue(s.SymbolNumber("varLengthMatrizL", i, 1))
      ColaServidorNumber = s.SymbolNumber("ColaServidor" & i)
      NumberInColaServidor = s.QueueNumberOfEntities(ColaServidorNumber)
      '... Si hay algún procesador disponible y trabajos en cola, se procede a lanzar el trabajo
      '... Antes de enviarlo al servidor, se comprueba que puede ser atendido dentro del plazo solicitado
      While (CapacidadLibre > 0) And (LengthMatrizL > 0) And (NumberInColaServidor > 0)
          '... Leemos el identificador del trabajo que queremos lanzar
          atTrabajoId = s.VariableArrayValue(s.SymbolNumber("VarMatrizColaL", i, 1))
          '... Identificamos la solicitud buscando la entidad en la cola principal
          '... en base al atributo atTrabajoId
          Call LocalizarEntidadC(EntidadSeleccionada, atTrabajoId, ColaServidorNumber)
          '... Quitamos el trabajo de la cola local
          s.QueueRemoveEntity EntidadSeleccionada, ColaServidorNumber
          NumberInColaServidor = s.QueueNumberOfEntities(ColaServidorNumber)
          '... Quitamos el trabajo de la matrizL correspondiente
          LengthMatrizL = s.VariableArrayValue(s.SymbolNumber("varLengthMatrizL", i, 1))
          Call EliminarTrabajoMatriz(EntidadSeleccionada, "VarMatrizColaL", LengthMatrizL, i)
          s.VariableArrayValue(s.SymbolNumber("varLengthMatrizL", i, 1)) = LengthMatrizL
```

```
            '... Guardamos en los atributos de la entidad, su duracion estimada y el momento de tiempo
            '... en que es enviada al servidor para su ejecución 'atTstart')
            Call CalculaEsperanzaC(EntidadSeleccionada, i, DuracionEstimada) ' Se usará a nivel estadístico
            s.EntityAttribute(EntidadSeleccionada, s.SymbolNumber("atTestimado")) = DuracionEstimada
            s.EntityAttribute(EntidadSeleccionada, s.SymbolNumber("atTstart")) = s.RunCurrentTime
            '... Verificamos si la tarea que se quiere asignar se puede atender con tiempo suficente
            Nollega = False
            Call CompruebaSiLlegaC(EntidadSeleccionada, Nollega, i)
            If Nollega = False Then
               '... Enviamos el trabajo a la estacion de entrada de los servidores para su ejecución
               '... Antes, actualizamos los datos de control del procesador en que se va a ejecutar
               '... el trabajo. Se utilizarán para las comprobaciones de viabilidad de planificación
               j = 1
               While (s.VariableArrayValue(s.SymbolNumber("varDdTareasEnCurso", i, j)) <> 0) And j <=
varNumCores
                  j = j + 1
               Wend
               s.EntityAttribute(EntidadSeleccionada, s.SymbolNumber("atNumCore")) = j
               Tfck = s.RunCurrentTime + DuracionEstimada
               s.VariableArrayValue(s.SymbolNumber("varDdTareasEnCurso", i, j)) = Tfck
               CapacidadLibre = CapacidadLibre - 1
               s.EntitySendToStation EntidadSeleccionada, estacionServidores
            End If
      Wend
   Next i

End Sub

Sub BuscarServRapidoEST(Entidad As Long)

   '   Rutina que busca la mejor opción para la planificación de un trabajo
   '   Comprueba cual es la opción que ofrece la entrega en un tiempo menor
   '   y comprueba igualmente que la planificación resultante de incluir
   '   el trabajo en cuestión sea viable

   Dim m As Model
   Dim s As Arena.SIMAN

   Dim i As Integer, j As Integer, k As Integer, n As Integer, kinsert As Integer
   Dim ExistePlanificacion As Boolean, PrimerCore As Boolean
   Dim EntidadTemporal As Long, Entidad0 As Long
   Dim Deadlinek As Long, DeadlineTarget As Long, DeadlineFirst As Long
   Dim varNumservidores As Integer, ServidorNumber As Long, varNumCores As Integer
   Dim ServMasRapido As Integer, PosServMasRapido As Integer, TfkServMasRapido As Double
   Dim DuracionEstimada As Double, T0 As Double, T0f As Double, Tfk As Long
   Dim ESTk As Double, DuracionEstimadak As Double, Tfkcalc As Long
   Dim LengthMatrizCi As Integer, atTrabajoId As Long
   Dim FirstCoreFree As Integer, PrimerServ As Boolean, ubicacionencontrada As Boolean
   Dim Tfcalc As Long, Tfck As Long, Tfc0 As Long
   Dim Core As Integer

   Dim ColaServidorNumber As Long, NumberInColaServidor As Integer

   Set m = ThisDocument.Model
   Set s = m.SIMAN

   '... Inicialización de variables
   varNumservidores = s.VariableArrayValue(s.SymbolNumber("varNumServidores"))
   varNumCores = s.VariableArrayValue(s.SymbolNumber("varNumCores"))
```

```
  DeadlineTarget = s.EntityAttribute(Entidad, s.SymbolNumber("atTlimite"))
  ExistePlanificacion = False

  '... Verificación para el caso en que no hay cola en el servidor

  For i = 1 To varNumservidores
      LengthMatrizCi = s.VariableArrayValue(s.SymbolNumber("varLengthMatrizL", i, 1))
      ColaServidorNumber = s.SymbolNumber("ColaServidor" & i)
      NumberInColaServidor = s.QueueNumberOfEntities(ColaServidorNumber)
      If LengthMatrizCi <> NumberInColaServidor Then GoTo SigServidorVacio ' Chequeo de seguridad
         ServidorNumber = s.SymbolNumber("RServidor" & i)
         If LengthMatrizCi = 0 And ExistePlanificacion = False Then
            '... No existe cola en el servidor, verificamos si hay algún procesador libre
            If (s.ResourceCapacity(ServidorNumber) - s.ResourceNumberBusy(ServidorNumber) > 0) Then
               s.EntityAttribute(Entidad, s.SymbolNumber("atNumServidor")) = i
               s.EntityAttribute(Entidad, s.SymbolNumber("atPosServidor")) = 1
               s.EntityAttribute(Entidad, s.SymbolNumber("atTipoAsignacion")) = 100 ' Parámetro de control
informativo
               ExistePlanificacion = True
            Else
               '... No hay cola ni ningún procesador libre
               '... Hay que buscar la tarea que acaba antes en todos los procesadores del servidor
               FirstCoreFree = 1
               DeadlineFirst = s.VariableArrayValue(s.SymbolNumber("varDdTareasEnCurso", i, 1))
               For j = 1 To varNumCores
                  Deadlinek = s.VariableArrayValue(s.SymbolNumber("varDdTareasEnCurso", i, j))
                  If Deadlinek < DeadlineFirst Then
                     FirstCoreFree = i
                     DeadlineFirst = Deadlinek
                  End If
               Next j
               '... Calculamos la duración estimada del trabajo a planificar
               Call CalculaEsperanzaC(Entidad, i, DuracionEstimada)
               '... Comprobar si el procesador que quedará libre podría entregar el trabajo
               '... a tiempo. NOTA: no comparamos con el otro servidor
               If DeadlineFirst < DeadlineTarget - DuracionEstimada Then
                  s.EntityAttribute(Entidad, s.SymbolNumber("atNumServidor")) = i
                  s.EntityAttribute(Entidad, s.SymbolNumber("atPosServidor")) = 1
                  s.EntityAttribute(Entidad, s.SymbolNumber("atTipoAsignacion")) = 200 ' Parámetro de control
informativo
                  ExistePlanificacion = True
               End If
            End If
         End If
SigServidorVacio:
  Next i

  If ExistePlanificacion = True Then GoTo LastLine

  '... Comienzo el código para el caso en que hay cola en el servidor i

  PrimerServ = True ' Se inicializa para controlar el servidor que entregará el trabajo más pronto
  For i = 1 To varNumservidores
     '... Inicialización de variables
     ServidorNumber = s.SymbolNumber("RServidor" & i)
     ColaServidorNumber = s.SymbolNumber("ColaServidor" & i)
     NumberInColaServidor = s.QueueNumberOfEntities(ColaServidorNumber)
     LengthMatrizCi = s.VariableArrayValue(s.SymbolNumber("varLengthMatrizL", i, 1))
     If (LengthMatrizCi <> NumberInColaServidor) Then GoTo SigServidor ' Chequeo de seguridad
```

```
    'Inicializamos el vector con los trabajos en curso en los procesadores
    For n = 1 To varNumCores
        s.VariableArrayValue(s.SymbolNumber("varDdCargaVirtual", i, n)) =
s.VariableArrayValue(s.SymbolNumber("varDdTareasEnCurso", i, n))
    Next n

    If NumberInColaServidor > varNumCores - 1 Then GoTo SigServidor ' Para intentar compensar cargas de colas

    ubicacionencontrada = False
    Deadlinek = 0
    k = 1 ' Contador para los elementos de la cola i

    '... Debemos considerar el caso en que haya un procesador disponible y el trabajo en estudio
    '... pueda tener mayor prioridad que el primero de la cola correspondiente

    If s.ResourceCapacity(ServidorNumber) - s.ResourceNumberBusy(ServidorNumber) <> 0 Then
        For n = 1 To varNumCores
            If s.VariableArrayValue(s.SymbolNumber("varDdCargaVirtual", i, n)) = 0 Then
                atTrabajoId = s.VariableArrayValue(s.SymbolNumber("varMatrizColaL", i, k))
                Call LocalizarEntidadC(EntidadTemporal, atTrabajoId, ColaServidorNumber)
                Deadlinek = s.EntityAttribute(EntidadTemporal, s.SymbolNumber("atTlimite"))
                If Deadlinek < DeadlineTarget Or ubicacionencontrada = True Then
                    Call CalculaEsperanzaC(EntidadTemporal, i, DuracionEstimadak)
                    k = k + 1
                Else
                    If ubicacionencontrada = False Then
                        Call CalculaEsperanzaC(Entidad, i, DuracionEstimadak)
                        Tfcalc = s.RunCurrentTime + DuracionEstimadak
                        ubicacionencontrada = True
                        kinsert = k
                    End If
                End If
            End If
            Tfck = s.RunCurrentTime + DuracionEstimadak
            s.VariableArrayValue(s.SymbolNumber("varDdCargaVirtual", i, n)) = Tfck
            '... Comprobamos que la planificación sigue siendo viable. En caso contrario
            '... pasamos al siguiente servidor para realizar el estudio de ubicación
            If Deadlinek < Tfck Then GoTo SigServidor
        Next n
    End If

    '... Preparamos las variables para seguir buscando una posición en cola o para
    '... verificar que la propuesta de ubicación en cola es una planificación viable
    Call BuscarNextTSaliente(Tfc0, i, Core)
    Tfck = Tfc0

    '... Si nuestro trabajo entra antes que la primera tarea en cola, pasamos a
    '... verificar si la planificación obtenida es viable
    If ubicacionencontrada = True Then GoTo VerificarSchedule

    '... Si hay cola en el servidor y todos los procesadores están ocupados
    '... localizamos la posición en cola que le correspondería a la nueva tarea en
    '... función de la prioridad EDF. Saldremos del bucle cuando hayamos encontrado una
    '... tarea menos prioritaria que la nuestra o hayamos alcanzado el final de la cola
    While (Deadlinek <= DeadlineTarget) And (k <= LengthMatrizCi)
        s.VariableArrayValue(s.SymbolNumber("varDdCargaVirtual", i, Core)) = Tfck
        Call BuscarNextTSaliente(Tfc0, i, Core)
        atTrabajoId = s.VariableArrayValue(s.SymbolNumber("varMatrizColaL", i, k))
```

```
        Call LocalizarEntidadC(EntidadTemporal, atTrabajoId, ColaServidorNumber)
        Call CalculaEsperanzaC(EntidadTemporal, i, DuracionEstimadak)
        Tfck = Tfc0 + DuracionEstimadak
        Deadlinek = s.EntityAttribute(EntidadTemporal, s.SymbolNumber("atTlimite"))
        k = k + 1
    Wend
    k = k - 1 ' Posiciono el contador en el último elemento de cola revisado
    If Deadlinek >= DeadlineTarget Then ' Hemos encontrado un hueco para la tarea a planificar
      '... Verificamos si se cumple su deadline(target) y el del resto de tareas que quedan en la cola
      Call CalculaEsperanzaC(Entidad, i, DuracionEstimada)
      Tfcalc = Tfc0 + DuracionEstimada
      kinsert = k
      Tfck = Tfcalc
VerificarSchedule:
      If Tfcalc < DeadlineTarget Then
        '... La tarea podría ser planificable. Hay que verificar el resto de cola a continuación
        While (Deadlinek >= Tfck) And (k <= LengthMatrizCi)
        s.VariableArrayValue(s.SymbolNumber("varDdCargaVirtual", i, Core)) = Tfck
        Call BuscarNextTSaliente(Tfc0, i, Core)
        atTrabajoId = s.VariableArrayValue(s.SymbolNumber("VarMatrizColaL", i, k))
        Call LocalizarEntidadC(EntidadTemporal, atTrabajoId, ColaServidorNumber)
        Call CalculaEsperanzaC(EntidadTemporal, i, DuracionEstimadak)
        Deadlinek = s.EntityAttribute(EntidadTemporal, s.SymbolNumber("atTlimite"))
        Tfck = Tfc0 + DuracionEstimadak
        k = k + 1
        Wend
      End If
      '... Verficamos bajo qué condiciones se sale del bucle para confirmar
      '... si la cola es realmente planificable o no lo es
      If (Deadlinek >= Tfck) And (k > LengthMatrizCi) Then
        '... Cola Planificable: Registramos que existe planificación y comprobamos
        '... si es la que nos entrega el trabajo antes
        ExistePlanificacion = True
        If PrimerServ = True Then
          ServMasRapido = i
          PosServMasRapido = kinsert
          TfkServMasRapido = Tfkcalc
          PrimerServ = False
        Else
          If Tfcalc < TfkServMasRapido Then
            ServMasRapido = i
            PosServMasRapido = kinsert
            TfkServMasRapido = Tfkcalc
          End If
        End If
      End If
    End If
SigServidor:
Next i

  If ExistePlanificacion = True Then
    s.EntityAttribute(Entidad, s.SymbolNumber("atNumServidor")) = ServMasRapido
    s.EntityAttribute(Entidad, s.SymbolNumber("atPosServidor")) = PosServMasRapido
    s.EntityAttribute(Entidad, s.SymbolNumber("atTipoAsignacion")) = 300
  End If

LastLine:
End Sub
```

```
Sub BuscarNextTSaliente(Tfc As Long, i As Integer, Core As Integer)

  '... Rutina para calcular cual es el procesador que va a finalizar su carga
  '... de trabajo en primer lugar, y cual es el instante de tiempo previsto
  '... en el que se liberará ... 'Core' + 'Tfc'

  Dim m As Model
  Dim s As Arena.SIMAN
  Dim j As Integer
  Dim varNumCores As Integer

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  varNumCores = s.VariableArrayValue(s.SymbolNumber("varNumCores"))
  Tfc = s.VariableArrayValue(s.SymbolNumber("varDdCargaVirtual", i, 1))
  Core = 1
  For j = 1 To varNumCores
     If s.VariableArrayValue(s.SymbolNumber("varDdCargaVirtual", i, j)) < Tfc Then
        Tfc = s.VariableArrayValue(s.SymbolNumber("varDdCargaVirtual", i, j))
        Core = j
     End If
  Next j

End Sub
```

PROCEDIMIENTOS AUXILIARES:

Los procedimientos principales anteriores para planificación **MYOPIC** hacen uso de unos procedimientos auxiliares que se exponen a continuación. Las funciones de estos procedimientos auxiliares son:

1. Actualizar la ocupación de procesadores a medida que van finalizando los trabajos.
2. Añadir un elemento en las matrices de colas ordenadas en función de priorirdad EDF.
3. Eliminar un componente de las matrices de colas y que se conserve la matriz ordenada y sin huecos.
4. Localizar los parámetros que definen una tarea dada, en base al atributo TrabajoID.
5. Estimar si un trabajo va a ser entregado a tiempo, antes de enviarlo a cola local.
6. Estimar si un trabajo va a ser entregado a tiempo antes de enviarlo al servidor.
7. Estimar la duración de un trabajo en un servidor definido.

```
Sub OLT3_B2F()

  '   Evento asociado a la finalización de la ejecución de un trabajo
  '   Se actualiza la ocupación de los procesadores indicando
  '   que el que estaba ocupado por el trabajo finalizado queda libre

  Dim m As Model
  Dim s As Arena.SIMAN
  Dim i As Integer, j As Integer
```

```
  Set m = ThisDocument.Model
  Set s = m.SIMAN

  i = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("atNumServidor"))
  j = s.EntityAttribute(s.ActiveEntity, s.SymbolNumber("atNumCore"))

  s.VariableArrayValue(s.SymbolNumber("varDdTareasEnCurso", i, j)) = 0

End Sub

Sub AnadirTrabajoMatriz(Entidad As Long, Varmatriz As String, LengthMatriz As Integer, ColaNumber As
Long, colaLength As Integer, line As Integer)

  '   Añadimos una solicitud de trabajo en matriz, respetando la priorización por EDF
  '   La matriz contendrá los atributos atTrabajoId de cada entidad para ser localizadas
  '   posteriormente

  Dim m As Model
  Dim s As Arena.SIMAN
  Dim i As Integer, k As Integer, LengthTemp As Integer
  Dim encontrado As Boolean
  Dim atTrabajoIdAux As Long, atTrabajoId As Long, Deadlinei As Long, DeadlineTarget As Long
  Dim Entidadi As Long

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  atTrabajoId = s.EntityAttribute(Entidad, s.SymbolNumber("atTrabajoId"))
  LengthTemp = LengthMatriz

  Select Case LengthTemp
   Case 0 '... Incorporar entidad activa
      s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, 1)) = atTrabajoId
      LengthMatriz = LengthMatriz + 1
   Case Else '... Incorporar entidad activa
      '... Verificar que no está en la matriz (double-check)
      encontrado = False
      i = 1
      While (i <= LengthMatriz And encontrado = False)
         atTrabajoIdAux = s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, i))
         If atTrabajoIdAux = atTrabajoId Then
            encontrado = True
         End If
      i = i + 1
      Wend

      If encontrado = False Then ' Añadir entidad a a matriz
         '... Se procede a buscar el punto de inserción para que la matriz quede ordenada por EDF
         i = 1
         DeadlineTarget = s.EntityAttribute(Entidad, s.SymbolNumber("atTlimite"))
         atTrabajoIdAux = s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, i))
         '... Localizamos la entidad (Entidadi) asociada al atributo atTrabajoIdAux
         Call LocalizarEntidadC(Entidadi, atTrabajoIdAux, ColaNumber)
         Deadlinei = s.EntityAttribute(Entidadi, s.SymbolNumber("atTlimite"))
         If LengthMatriz > 1 Then ' sigo buscando el punto de inserción
            '... Desplazamos 'i' hasta encontrar el punto de inserción con criterio EDF
            While (Deadlinei <= DeadlineTarget) And (i < LengthMatriz)
                  i = i + 1
                  atTrabajoIdAux = s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, i))
```

```
                '... Localizamos la entidad (Entidadi) asociada al atributo atTrabajoIdAux
                Call LocalizarEntidadC(Entidadi, atTrabajoIdAux, ColaNumber)
                Deadlinei = s.EntityAttribute(Entidadi, s.SymbolNumber("atTlimite"))
            Wend
            k = LengthMatriz + 1
            If (DeadlineTarget < Deadlinei) Then
              '... Hemos encontrado un hueco en la matriz para nuestra nueva entidad
              While k > i ' Hacemos hueco en la matriz para incluir la Entidad nueva
                atTrabajoIdAux = s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, k - 1))
                s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, k)) = atTrabajoIdAux
                k = k - 1
              Wend
              '... Añadimos la entidad nueva en su ubicación
              s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, i)) = atTrabajoId
            Else
              '... No hay hueco sino que hemos llegado al final de la matriz
              s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, k)) = atTrabajoId
            End If
          Else ' en el caso que la matriz sólo tenga una entidad
            If (Deadlinei <= DeadlineTarget) Then
              s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, 2)) = atTrabajoId
            Else
              s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, 2)) = atTrabajoIdAux
              s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, 1)) = atTrabajoId
            End If
          End If
          LengthMatriz = LengthMatriz + 1
        End If
    End Select
    s.EntityAttribute(Entidad, s.SymbolNumber("atTentmatriz")) = s.RunCurrentTime
    s.EntityAttribute(Entidad, s.SymbolNumber("atLmatriz")) = LengthMatriz
End Sub

Sub EliminarTrabajoMatriz(Entidad As Long, Varmatriz As String, LengthMatriz As Integer, line As Integer)

  '   Eliminar una solicitud de trabajo de la matriz
  '   Se localiza la posición en la matriz a través del atributo atTrabajoId

  Dim m As Model
  Dim s As Arena.SIMAN
  Dim i As Integer, k As Integer
  Dim encontrado As Boolean
  Dim atTrabajoIdAux As Long, atTrabajoId As Long

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  i = 0
  encontrado = False
  '... Incrementamos 'i' mientras no hayamos encontrado la solicitud de trabajo en la matriz
  While (i <= LengthMatriz And encontrado = False)
      i = i + 1
      atTrabajoIdAux = s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, i))
      If s.EntityAttribute(Entidad, s.SymbolNumber("atTrabajoID")) = atTrabajoIdAux Then
          encontrado = True
      End If
  Wend
  If encontrado = True Then
      '... Eliminar entidad en matriz y reubicar el resto de posiciones hasta el final
```

```
    '... Guardamos el valor de 'i' en 'atPosMatriz' para verificaciones de control
    '... En un funcionamiento normal, el valor de 'i' debería ser 1
    s.EntityAttribute(Entidad, s.SymbolNumber("atPosMatriz")) = i
    While (i < LengthMatriz)
      atTrabajoIdAux = s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, i + 1))
      s.VariableArrayValue(s.SymbolNumber(Varmatriz, line, i)) = atTrabajoIdAux
      i = i + 1
    Wend
  LengthMatriz = LengthMatriz - 1
  End If

End Sub

Sub LocalizarEntidadC(Entidad As Long, atTrabajoId As Long, ColaNumber As Long)

  '... Se recibe un valor de atributo 'atTrabajoId' y un identificador de cola
  '... 'ColaNumber' para buscar la entidad correspondiente en dicha cola y
  '... ser devuelta a través de 'Entidad'

  Dim m As Model
  Dim s As Arena.SIMAN
  Dim i As Integer
  Dim atTrabajoIdAux As Long
  Dim colaLength As Integer

  Set m = ThisDocument.Model
  Set s = m.SIMAN

  '... Inicialización de valores para iniciar la búsqueda
  i = 1
  colaLength = s.QueueNumberOfEntities(ColaNumber)
  atTrabajoIdAux = s.QueuedEntityAttribute(ColaNumber, i, s.SymbolNumber("atTrabajoId"))
  '... Incremento de 'i' mientras no se alcance la entidad cuyo atTrabajoId
  '... corresponda con el solicitado a la entrada en este procedimiento
  While ((i < colaLength) And (atTrabajoId <> atTrabajoIdAux))
    i = i + 1
    atTrabajoIdAux = s.QueuedEntityAttribute(ColaNumber, i, s.SymbolNumber("atTrabajoId"))
  Wend
  '... Se guarda la entidad en la variable 'Entidad'
  Entidad = s.QueueEntityLocationAtRank(i, ColaNumber)

End Sub
```

Además de los procedimientos auxiliares enumerados más arriba, el algoritmo Myopic hace uso también de procedimientos ya existentes para EDF y LLF, que han sido presentados en el aparado B.1 y B.2. Éstos son:

Sub **CompruebaSiLlegaP(Entidad As Long, NollegaP As Boolean)**

Sub **CompruebaSiLlegaC(Entidad As Long, Nollega As Boolean, ServidorEscogido As Integer)**

Sub **CalculaEsperanzaC(Entidad As Long, Servidor As Integer, DuracionEstimada As Double)**

ANEXO C: Tablas de datos extraídos de las simulaciones

En este anexo se exponen las tablas con la recogida de los datos realizada con la simulación, para cada tipo de planificación estudiada.

Hay que notar que la última fila de cada tabla, identificada como servidor '3' representa en realidad la recogida del número de trabajos expulsados antes de ser enviados a las colas locales de los servidores.

	PLANIFICACIÓN EDF										
Servidor	**Tipo de Trabajo**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	Earliness Avg (min)	14,6083	999,7839	842,0683	168,1070	978,6013	14,1606	17,5083	177,7012	10079,8972	1439,7826
2	Earliness Avg (min)	16,4183	1046,8385	847,0161	165,5311	994,0594	13,2284	17,8924	177,7082	10079,8965	1439,7809
1	Lateness Avg (min)	40,1250	0	0	0	0	0	3,5363	0	0	0
2	Lateness Avg (min)	3,1201	0	0	0	0	3,5385	0	0	0	0
1	Trabajos Procesados (ud)	16	143	48	45	53	9	183	140	2694	21
2	Trabajos Procesados (ud)	18	104	51	47	44	10	132	84	1728	48
1	Entregas en Plazo (ud)	12	143	48	45	53	9	182	140	2694	21
2	Entregas en Plazo (ud)	16	104	51	47	44	9	132	84	1728	48
1	Entregas fuera Plazo (ud)	4	0	0	0	0	0	1	0	0	0
2	Entregas fuera Plazo (ud)	2	0	0	0	0	1	0	0	0	0
1	Peticiones expulsadas del sistema (ud)	2	0	0	0	0	0	0	0	0	0
2	Peticiones expulsadas del sistema (ud)	0	0	0	0	0	0	0	0	0	0
3	Peticiones expulsadas del sistema (ud)	0	0	0	0	0	0	0	0	0	0

Tabla C.1

	PLANIFICACIÓN LLF										
Servidor	**Tipo de Trabajo**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	Earliness Avg (min)	16,0810	1030,4483	844,7797	167,0568	968,8968	15,9383	17,7481	176,6993	10079,8972	1439,7936
2	Earliness Avg (min)	15,4223	1013,0288	843,9473	166,6032	953,6666	13,3703	17,5332	177,4751	10079,8966	1439,7798
1	Lateness Avg (min)	32,5858	0	0	0	0	0	4,3921	0	0	0
2	Lateness Avg (min)	4,1214	0	0	0	0	3,5385	0	0	0	0
1	Trabajos Procesados (ud)	22	118	46	45	56	7	178	130	2630	8
2	Trabajos Procesados (ud)	14	129	53	47	41	12	137	94	1792	61
1	Entregas en Plazo (ud)	17	118	46	45	56	7	177	130	2630	8
2	Entregas en Plazo (ud)	13	129	53	47	41	11	137	94	1792	61
1	Entregas fuera Plazo (ud)	5	0	0	0	0	0	1	0	0	0
2	Entregas fuera Plazo (ud)	1	0	0	0	0	1	0	0	0	0
1	Peticiones expulsadas del sistema (ud)	0	0	0	0	0	0	0	0	0	0
2	Peticiones expulsadas del sistema (ud)	0	0	0	0	0	0	0	0	0	0
3	Peticiones expulsadas del sistema (ud)	0	0	0	0	0	0	0	0	0	0

Tabla C.2

	PLANIFICACIÓN MYOPIC										
Servidor	**Tipo de Trabajo**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	Earliness Avg (min)	15,7992	1040,8504	851,1767	166,0534	1010,6876	13,0331	16,7741	178,2467	10079,6793	1439,7642
2	Earliness Avg (min)	13,7021	992,6766	833,7564	166,1500	947,2579	13,9705	17,2037	175,5875	10079,8964	1439,7748
1	Lateness Avg (min)	42,5746	0	0	0	0	3,53854	3,9763	0	0	0
2	Lateness Avg (min)	3,2841	0	0	0	0	0,0000	0	0	0	0
1	Trabajos Procesados (ud)	27	152	52	48	69	11	231	131	4412	62
2	Trabajos Procesados (ud)	9	95	47	44	28	8	82	93	10	7
1	Entregas en Plazo (ud)	23	152	52	48	69	10	230	131	4412	62
2	Entregas en Plazo (ud)	7	95	47	44	28	8	82	93	10	7
1	Entregas fuera Plazo (ud)	4	0	0	0	0	1	1	0	0	0
2	Entregas fuera Plazo (ud)	2	0	0	0	0	0	0	0	0	0
1	Peticiones expulsadas del sistema (ud)	0	0	0	0	0	0	0	0	0	0
2	Peticiones expulsadas del sistema (ud)	0	0	0	0	0	0	0	0	0	0
3	Peticiones expulsadas del sistema (ud)	0	0	0	0	0	0	0	0	0	0

Tabla C.3